Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Refatoramento da Body Sensor Network (BSN) para detecção de casos suspeitos de covid-19

Gabriel R. Pacheco

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientadora
Prof.a Dr.a Genaína Nunes Rodrigues

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Refatoramento da Body Sensor Network (BSN) para detecção de casos suspeitos de covid-19

Gabriel R. Pacheco

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof.a Dr.a Genaína Nunes Rodrigues (Orientadora)
CIC/UnB

Prof. Dr. Daniel de Paula Porto
Universidade de Brasília

Prof. Dr. Vinícius Ruela Pereira Borges
Universidade de Brasília

Prof. Dr. Jorge Henrique Cabral Fernandes
Coordenador do Curso de Computação — Licenciatura

Brasília, 27 de dezembro de 2023

Dedicatória

Dedico esse trabalho aos meus pais, Valéria e Marcelo, e ao meu irmão, Víctor, por todo apoio, amor e suporte ao longo de todo esse percurso;

Não posso esquecer também dos meus sócios e grandes amigos Rafael Mascarenhas, Mohana Kruger e, mais uma vez, Víctor Pacheco, por toda a parceria, inspiração e aprendizado;

Aos meus amigos, que longe ou perto estiveram presentes nessa jornada, com destaque para a Clara, o Matheus, o Caio e o Rodrigo;

Ao movimento escoteiros, que teve importante impacto na minha formação e em quem sou hoje. Aqui conheci pessoas maravilhosas e colecionei memórias, tanto como jovem quanto como voluntário.

Agradecimentos

O presente trabalho se desenvolveu com apoio do Laboratório de Engenharia de *Software* (LES), envolvendo meus colegas e orientadores. Agradeço a Universidade de Brasília (UnB) por todas as contribuições a este estudo e à minha formação;

Agradeço imensamente aos meus colegas de laboratório e pesquisa, com carinho especial ao Ricardo Diniz Caldas e ao Carlos Eduardo Taborda Lottermann, sem eles esse trabalho não existiria. Também um agradecimento ao Vicente Romeiro De Moraes que ajudou a superar os obstáculos do caminho;

Aos bons professores que me guiaram, tanto na escola quanto na faculdade, especialmente minha orientadora, Prof.a Dr.a Genaína Nunes Rodrigues por toda confiança e suporte no decorrer deste trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Considerando um cenário pós pandêmico marcado pela era da tecnologia e informação, esse trabalho foca no aperfeiçoamento da *Body Sensor Network* (BSN), ou, em português, Rede de Sensores Corporais. De forma geral, a BSN foi concebida para funcionar dentro e fora das instituições de saúde, permitindo triagem e detecção de emergências e riscos em tempo real. O projeto visa integrar as áreas das tecnologias de informação com as de saúde, oferecendo uma solução para os atuais e futuros desafios enfrentados, especialmente durante crises como a da pandemia de covid-19.

Essa monografia busca desenvolver e aprimorar um sistema para detectar casos suspeitos de covid-19, focando em duas frentes para essa compatibilidade. Na primeira implementa-se novas interfaces de coleta de dados e na segunda desenvolve-se o algoritmo de Aprendizado de Máquina (AM) para a detecção desses casos. Ao falar da coleta de dados, abre-se três possibilidades de entradas para alguns dos seus sensores virtuais: dados gerados a partir de simulação, dados documentados tabelados inseridos pelo operador e dados reais coletados do paciente por meio de sensores físicos. Na segunda frente empreende-se o uso de um algoritmo de aprendizado supervisionado do tipo Máquina de Vetores de Suporte (SVM) para detectar esses casos suspeitos de covid-19 com base nos sinais fisiológicos de entrada, comparando os casos de entrada com os fornecidos como treinamento.

Palavras-chave: Rede de Sensores Corporais, BSN, SA-BSN, Body Sensor Network, Detecção de covid-19, Sistema de saúde, Sistemas Auto-adaptativos, SVM, covid-19

Abstract

Considering a post-pandemic scenario marked by the era of technology and information, this work focuses on improving the Body Sensor Network (BSN). Overall, the BSN is designed to work inside and outside healthcare institutions, enabling real-time screening and detection of emergencies and risks. The project aims to integrate the areas of information technologies and health, offering a solution to current and future challenges faced, especially during crises such as the COVID-19 pandemic.

This monograph seeks to develop and improve a system to detect suspect cases of COVID-19, focusing on two fronts for this mission. In the first one, new data collection interfaces were implemented and in the second one, the Machine Learning (ML) algorithm was developed to detect these cases. When talking about data collection, three input possibilities were open up for some of its virtual sensors: data generated from simulation, tabulated documented data entered by the operator and real data obtained from the patient through physical sensors. On the second front, the use of a supervised learning algorithm of the Support Vector Machine (SVM) type is undertaken to detect these suspected cases of COVID-19 based on physiological input signals, comparing the input cases with those provided as training.

Keywords: Body Sensor Network, Self-Sdaptive Body Sensor Network, BSN, SA-BSN, COVID-19 detection, Health system, Self-Adaptive System, SVM, COVID-19

Sumário

Lista de Figuras

Lista de Tabelas

+

Lista de Abreviaturas e Siglas

AM *Machine Learning.*

BAN *Body Area Network.*

BPM Batimentos por Minuto.

BSN *Body Sensor Network.*

CSV Valores Separados por Vírgula.

ECG Eletrocardiograma.

EPI Equipamento de Proteção Individual.

GPIO *General Purpose Input/Output.*

IDE Ambiente de Desenvolvimento Integrado.

IoT *Internet* das Coisas.

LES Laboratório de Engenharia de Software.

NB *Naïve Bayes.*

RBF *Radial Basis Function.*

RNAs Redes Neurais Artificiais.

ROS *Robot Operating System.*

SA-BSN *Self-Adaptive - Body Sensor Network.*

SARS-CoV-2 coronavírus da síndrome respiratória aguda grave 2.

SO Sistema Operacional.

SVC *Support Vector Classification.*

SVM *Suport Vector Machine.*

SVR *Support Vector Regression.*

TI Tecnologia da Informação.

UnB Universidade de Brasília.

USB *Universal Serial Bus.*

WBAN *Wireless Body Area Network.*

WBSN *Wireless Body Sensor Networks.*

Capítulo 1

Introdução

Covid-19 é o nome da doença provocada pelo coronavírus da síndrome respiratória aguda grave 2 (SARS-CoV-2). Essa expressão vem do acrônimo inglês COVID, proveniente de *co*rona*vi*rus *d*isease, e esse vírus contou com uma rápida disseminação [2][3]. A doença por coronavírus causou o maior desafio global e de saúde pública, após a pandemia do surto de gripe de 1918 [3]. Enfrentando esse cenário pós pandêmico marcado com uma permeabilidade crescente dos dispositivos da tecnologia da informação nos lares [4], vê-se uma nova realidade na qual é preciso incorporar essas tecnologias em diversas áreas de modo mais acessível.

Como proposta para esse processo, existe a *Body Sensor Network* (BSN) [5], ou, em português, Rede de Sensores Corpóreos, dotada de sistemas auto adaptáveis e um conjunto de sensores voltados para analisar e prever o estado do paciente, podendo esse sistema ser operado fora das instituições de saúde com capacidade de triagem e detecção de emergências e riscos [1]. Ao falar de *Body Sensor Network* (BSN) existem diversos pontos relevantes de pesquisa, estudos como o [6] que estuda a parte de alimentação desse tipo de tecnologia, o [5] que estuda o uso de sensores *touch* focado na acessibilidade para pessoas com deficiência física e o [7] que usa uma rede de sensores para estimar a posição do corpo do paciente.

Citando trabalhos correlatos no processo de detecção de casos suspeitos de covid-19, pode-se mencionar outros estudos que adotam análises de dados para a detecção da doença do corona vírus. Um trabalho interessante é o [8] que estuda a variabilidade do batimento cardíaco associada com casos infecção por covid-19 e observado sinais prematuros na identificação clínica usando o sensoriamento do dispositivo *weareble* Apple Watch. Outro estudo que vale a pena ser salientado é o [9] que usa uma abordagem de *deep learning* para análise de exames de imagem raio-X do torax para detecção de casos de covid-19 e pneumonia.

Esse trabalho baseia-se no estudo, análise e aprimoramento da BSN, tanto do ponto de

vista técnico de sua implementação, quanto de suas possibilidades como projeto integrador de áreas. Nesse projeto, o foco é a implementação da detecção de casos suspeitos de covid-19. Para isso, trabalha-se com dois pilares: o primeiro consiste em aprimorar o processo de entrada de dados a serem processados pela BSN, desenvolvendo novas possibilidades de testes e aplicações. Aqui se possibilita o uso de dados simulados, dados externos tabelados previamente gerados ou aferidos e dados reais coletados em tempo real de um paciente. No segundo pilar há o processamento desses dados usando um modelo de aprendizado supervisionado do tipo *Suport Vector Machine* (SVM), ou, em inglês, *Support vector machine* (SVM), usando como base os dados da entrada do paciente para detectar possíveis casos de covid-19.

1.1 Justificativa do trabalho

A situação global pós pandêmica trouxe à tona questões relevantes sobre os sistemas de saúde, em uma escala regional e global, assim como a sua capacidade de fornecer serviços médicos de qualidade. A adaptação e aumento da aceitação das redes de sensores, como a *Body Sensor Network*, demonstra a sua utilidade como ferramenta para viabilizar o tratamento remoto no acompanhamento clínico de pacientes com covid-19 ou outros ambientes, e, ao mesmo tempo que protege os médicos, enfermeiras e outros funcionários da saúde ao tratar os pacientes [10]

Dessa forma, este trabalho almeja aumentar a versatilidade e autonomia da BSN, adequando-a para detectar enfermidades em tempo real, o que pode ser de uma grande valia sob a ótica social. Com isso, almeja-se contribuir para reduzir a taxa de contágio de doenças e viabilizar um diagnóstico mais prematuro e de baixo custo. A *Body Sensor Network* (BSN) foi escolhida para esse trabalho pois ao longo do tempo essa tecnologia esta se tornando cada vez mais significativos e amplamente adotada [11]. Pelo seu custo acessível e capacidade de expansão com novos módulos, de forma simples e versátil, essa base já desenvolvida se mostrou propicia para o trabalho aqui enfrentado.

1.2 Objetivos Gerais e Específicos

Este trabalho tem como objetivo principal estender o projeto da *Body Sensor Network* (BSN) [1] para a detecção de casos suspeitos de covid-19. Para que isso seja possível, este trabalho apoia-se em dois principais pilares: o primeiro é viabilizar diversas fontes de entradas de dados para a BSN via dados simulados, dados externos (base de dados externa) e dados reais do paciente coletados com sensores físicos em tempo real. O segundo pilar consiste na implementação de um módulo de *Machine Learning* na BSN para a análise

e detecção de casos suspeitos de covid-19. Dessa forma, como objetivos específicos deste estudo, propõe-se:

1. Interfacear e selecionar possibilidades de entradas de dados na *Body Sensor Network* (BSN);
2. Aplicar um modelo de predição de casos suspeitos de covid-19 em tempo real com base em aprendizado de máquina usando os dados fisiológicos disponíveis em uma BSN genérica;
3. Testar confiabilidade da detecção de casos de covid-19 com base em um modelo de predição.

1.3 Questões de Pesquisa

A principal questão utilizada para nortear o desenvolvimento deste trabalho é: "Como estender a *Body Sensor Network* (BSN) [1] para detecção de casos suspeitos de covid-19 tanto em ambientes de simulação quanto reais (i.e. originários de sensores físicos) preservando sua arquitetura original?"

Tendo essa orientação em questão, tem-se como objetivo neste trabalho responder às seguintes indagações:

1. Qual é o papel da *Body Sensor Network* (BSN) na detecção de casos de covid-19?
2. Como especializar uma BSN para detecção de covid-19?
3. O que desenvolvedores de uma *Body Sensor Network* devem considerar na especialização de uma BSN genérica para detecção de covid-19?

1.4 Metodologia

Metodologias de desenvolvimento consistem em um conjunto sistemático de abordagens e processos para a criação de uma solução [12]. Nesse trabalho adota-se alguns conceitos da Ciência do Projeto, do inglês *Design Science* e uma breve revisão da literatura.

Para entender o funcionamento geral do projeto da BSN, há, neste estudo uma separação entre as organizações do hardware, e da camada de software. Ou seja, estuda-se primeiramente a estrutura física semimóvel que executa o projeto e realiza a interação com o ambiente e com o paciente e, posteriormente, explora-se os aspectos programacionais da estrutura e arquitetura de funcionamento computacional que sintetiza esses dados e fornece um conjunto de informações como resposta.

No que diz respeito à Seção 1.3 de Questões de Pesquisa, para primeira pergunta busca-se executar uma breve revisão de literatura, na qual estudam-se outros trabalhos da área. Para a segunda tem-se um levantamento dos requisitos necessários para a implantação das novas funcionalidades do software e, por último, na terceira questão, busca-se uma reflexão do trabalho efetuado e quais suas consequências. De forma mais abstrata, todos esses pontos serão abordados durante o desenvolvimento do artefato computacional proposto, sendo cada nova funcionalidade explicada, testada e debatida neste trabalho.

1.5 Organização do Trabalho

Os demais capítulos deste trabalho estão organizados da seguinte forma. No segundo capítulo são apresentadas as tecnologias e recursos adotados, assim como o referencial teórico que fundamenta este trabalho. No terceiro capítulo é apresentada em detalhe a contribuição deste trabalho quanto à nova disposição arquitetural da BSN e as implementações necessárias para estender permitir que a BSN possa prover contribuir na detecção de casos suspeitos de covid-19. O Capítulo 4 apresenta os resultados obtidos na avaliação da BSN estendida. E por fim, apresenta-se no Capítulo 5 as conclusões e sugestões para trabalhos futuros, assim como possíveis melhorias.

Capítulo 2

Referencial teórico e tecnologias adotadas

Como cerne do estudo desse trabalho, a *Body Sensor Network* (BSN) é explicada e melhor detalhada neste capítulo. Diversas tecnologias e conceitos importantes também são explanados junto de alguns dos aspectos fundamentais para a compreensão panorâmica do projeto. Eles são brevemente elencados a seguir.

2.1 *Robot Operating System* (ROS)

Para o controle geral e estruturação dos sistemas é utilizado o *Robot Operating System* (ROS), que é um *framework open source* utilizado, como o nome sugere, em projetos de robótica. Foi desenvolvido pelo laboratório *Willow Garage* e lançado em 2010. O ROS facilita o desenvolvimento de sistemas robóticos complexos, permitindo colaboração e compartilhamento de código. Suporta linguagens como C++ e Python, e possui bibliotecas e ferramentas para controle de movimento, percepção sensorial e comunicação. Em 2017 foi lançado o ROS 2, trazendo melhorias como suporte a sistemas distribuídos em tempo real. O *Robot Operating System* (ROS) é amplamente utilizado em aplicações acadêmicas, industriais e de pesquisa em todo o mundo[13]. Ele é usado como base de estruturação do projeto, servindo para criar os módulos/nós e possibilitando a comunicação entre eles. O funcionamento básico das estruturas adotadas é o seguinte:

- **Nó:** Um nó no ROS é um processo executável independente que realiza cálculos ou comunicações. Os nós são conectados uns aos outros por meio de estruturas chamadas tópicos [14].

- **Tópico:** É um canal de comunicação que permite que os nós publiquem e assinem mensagens. As mensagens são estruturas de dados que contêm informações que os nós podem usar para se comunicar entre si [14].

Para entender melhor esse conceito pode ser usado um exemplo: imagine que você tem um robô com um conjunto de motores e um sensor de distância. Você pode criar dois nós diferentes: um para processar o resultado do sensor de distância e outro para acionar os motores. Esses dois nós podem se comunicar por meio de tópicos, o nó do sensor pode publicar a distância aferida em um tópico e o nó do do motor pode assinar esse tópico para receber a medida.

2.2 Cadeia de Markov

Uma cadeia de Markov, do inglês *Markov Chain*, é um modelo matemático que descreve uma sequência de eventos, onde a probabilidade de cada evento depende apenas do estado do evento anterior [15]. Essa estrutura pode ser pensada como um sistema que transita entre diferentes estados ao longo do tempo. As cadeias de Markov são frequentemente representadas usando um gráfico direcionado, onde os nós representam os estados e as arestas representam as transições entre os estados [16]. Como pode-se observar Figura 2.1.

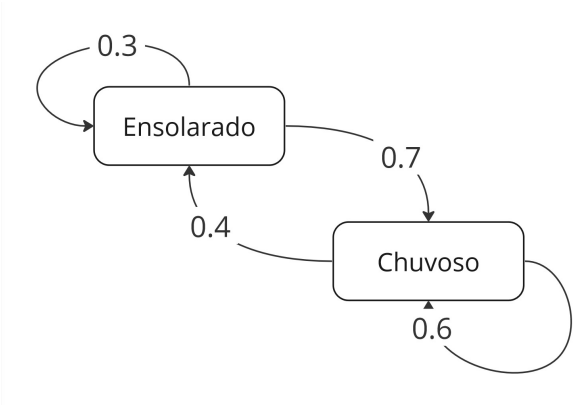


Figura 2.1: Gráfico representativo de uma cadeia de Markov para a probabilidade do dia ser Chuvoso ou ser Ensolarado.

Nesse exemplo, imagina-se que há um modelo meteorológico com dois estados, representados dentro de cada uma das caixas: ensolarado e chuvoso. As probabilidades de transição entre estados são mostradas nas setas, representando a chance de cada transição. Por exemplo, caso o estado atual seja ensolarado, há uma probabilidade de 30% de permanecer ensolarado e de 70% de transição para um período chuvoso.

Esse modelo é usado no projeto junto aos dados provenientes de simulação para calcular a previsão do estado do paciente usando valores nominais tabulados para cada sensor, prevendo qual será seu estado futuro com base em seu estado atual. No Listing 2.1 pode-se ver um exemplo de sua implementação. Na metade superior pode-se observar as probabilidades estimadas, em percentagem, de cada transição, e na outra metade os valores estabelecidos para cada faixa de risco.

Listing 2.1: Trecho da configuração do módulo do paciente. Cadeia de Markov implementada na BSN para representar as probabilidades de mudança de estado de risco, na parte superior, e as faixas de leituras esperada para cada segmento, no conjunto de linhas inferiores.

```
<!-- Markov chain for oxigenation -->
<param name="oxigenation_State0" value="0,0,0,0,0" />
<param name="oxigenation_State1" value="0,0,0,0,0" />
<param name="oxigenation_State2" value="0,0,33,33,34" />
<param name="oxigenation_State3" value="0,0,33,33,34" />
<param name="oxigenation_State4" value="0,0,33,33,34" />

<!-- Risk values for oximeter -->
<param name="oxigenation_HighRisk0" value="-1,-1"/>
<param name="oxigenation_MidRisk0" value="-1,-1"/>
<param name="oxigenation_LowRisk" value="65,100" />
<param name="oxigenation_MidRisk1" value="55,65" />
<param name="oxigenation_HighRisk1" value="0,55" />
```

2.3 Body Sensor Network (BSN)

A *Body Sensor Network* (BSN), também conhecida como *Body Area Network* (BAN), é uma tecnologia que interconecta vários sensores e nós atuadores sobre, dentro e ao redor do corpo humano [6]. Com a ajuda de sensores miniaturizados e atuadores, um sistema de BSN pode medir, analisar e comunicar sinais do corpo humano. Esses dados fisiológicos são a seguir transmitidos a uma unidade de processamento local, como um computador ou celular, ou a um serviço médico pessoal. Essa tecnologia também pode contar com uma versão sem fio, sendo essa denominada *Wireless Body Area Network* (WBAN) ou *Wireless Body Sensor Networks* (WBSN) [17].

Aqui será adotada a versão da *Body Sensor Network* (BSN) disponível na nota[1] que foi desenvolvida no laboratório Laboratório de Engenharia de Software (LES) da Universidade de Brasília (UnB). Tal trabalho é o mesmo adotado em outras pesquisas como [1] e [17], sendo aqui desenvolvido um novo conjunto de funcionalidades para contribuir com esse sistema.

Para entender melhor o complexo projeto da BSN, essa é separada em duas partes profundamente interligadas, o hardware e o software. Abaixo também existem as referências para replicação do projeto para facilitar seu uso e seu estudo.

2.3.1 Instalação do software e acesso aos arquivos

O guia com as etapas de como baixar e usar o software adotado e desenvolvido, está disponível na página do GitHub `https://github.com/lesunb/bsn_seams24`. É viável utilizá-lo com ou sem a leitura de dados reais de um paciente por meio de sensores físicos, sendo necessário para isso o uso do Arduino e dos próprios sensores físicos, o que torna possível tanto a análise de dados reais de um paciente quanto a utilização de dados provenientes de uma simulação virtual. No mesmo link se pode observar a organização completa do projeto, estando esse aberto e disponível a todos.

2.3.2 Organização de hardware

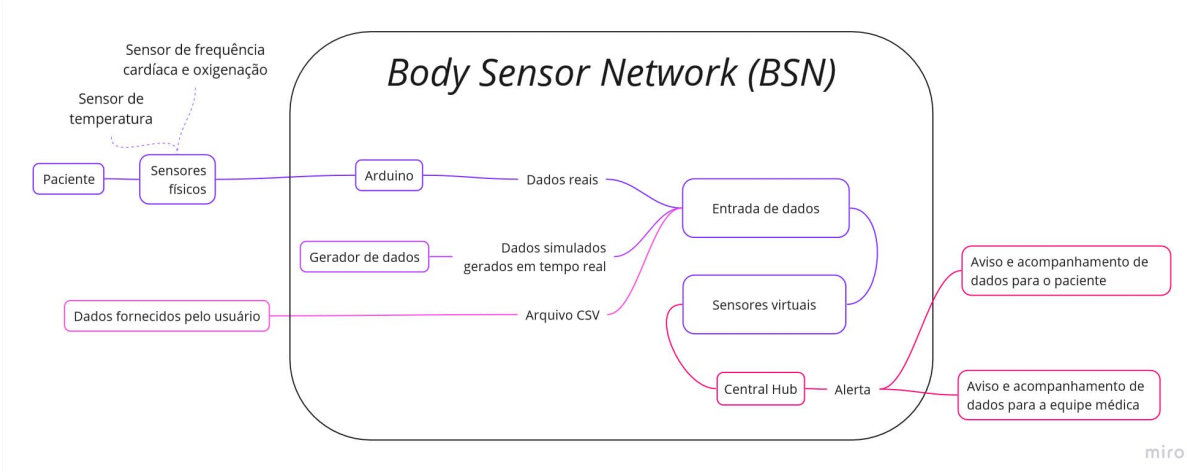


Figura 2.2: Representação da organização física básica da BSN com seus principais módulos e caminho básico de dados.

Na Figura 2.2 pode-se observar a relação entre a *Body Sensor Network* e o meio, bem como seus principais componentes. Nela existe uma representação da organização física

[1]Versão da BSN adotada: `https://github.com/lesunb/bsn_seams24`

das entradas de dados do projeto, apresentando o caminho do fornecimento das leituras. Nessa figura existe um grande enfoque no processo de entrada de dados, existindo três ramificações conectadas a esse bloco: O Arduino, responsável pelos dados reais provenientes dos Sensores físicos que são ligados ao Paciente; o Gerador de dados, que gera dados simulados em tempo real; e os Dados fornecidos pelo usuário, que são interfaceados por arquivos CSV fornecidos por ele. A entrada de dados é uma interface entre as opções de entradas e os Sensores virtuais, sendo esses Sensores virtuais conectados ao *Central Hub*. Voltando novamente a parte da esquerda da figura, existem opções de coleta de dados, onde o usuário pode escolher a fonte desejada entre uma delas. A função de cada uma delas está melhor descrita abaixo:

- Sensor físico: Usado para aferir os dados fisiológicos de um paciente em tempo real, esse sensor é interfaceado pelo Arduino e fornece dados reais a "Entrada de dados". Na atual versão existe o suporte a dois sensores físicos que aferem os dados do paciente, o sensor de temperatura e o sensor de frequência cardíaca e oxigenação.
- Gerador de dados: Essa coleta fornece dados simulados gerados em tempo real para a "Entrada de dados". Aqui é usada a Cadeia de Markov para definir as probabilidades na geração desses dados simulados pseudoaleatórios e essa é a entrada padrão da BSN, sendo a opção de backup caso alguma outra entrada falhe.
- Dados fornecidos pelo usuário: Nessa seleção de dados externos é usado o conjunto de dados tabelados e inseridos pelo operador e então fornecidos a "Entrada de dados". Aqui o usuário fornece um arquivo do tipo Valores Separados por Vírgula (CSV) para o sensor de entrada que deseja usar, e os dados inseridos nesse arquivo serão transmitidos a BSN quando a leitura do respectivo sensor virtual for requerida.

Essa entrada de dados é um dos pontos cruciais deste trabalho. Inicialmente apenas a entrada de dados gerados estava implementada e a proveniente de sensores físicos encontrava-se parcialmente desenvolvida. Seguindo o objetivo de analisar o estado de um sujeito, denominado paciente, é feita a coleta de uma série de leituras de seu estado fisiológico. O sistema conta com dois tipos diferentes de sensores, são eles:

- **Sensores virtuais:** São os que representam as entradas de dados da *Body Sensor Network* no *Central Hub*, assim como pode ser observado na Figura 2.5, se referindo ao dado que é aceito no programa de forma geral. A origem de dados para esses sensores podem ser ou dados gerados por simulação, ou dados externos inseridos pelo usuários, ou até mesmo dados reais coletados com auxílio de sensores físicos. Para cada sensor existe um mapeamento com um nó representado na Figura 2.6, mais precisamente na parte mais a direita da figura. São os sensores virtuais adotados nessa versão da ferramenta:

  - Eletrocardiograma (ecg_data): para frequência cardíaca e curva eletrocardiográfica. Esse sensor virtual está implementado dentro do nó "/G3T1_2";
  - Oxímetro de pulso (oximeter_data): para determinar a saturação de oxigênio no sangue. Esse sensor virtual está implementado dentro do nó "/G3T1_1";
  - Termômetro (thermometer_data): que coleta a temperatura corporal em graus Celsius. Esse sensor virtual está implementado dentro do nó "/G3T1_3",
  - Esfigmomanômetro (abps_data e abpd_data): para medir a pressão arterial sistólica e a pressão arterial diastólica. Esse sensor virtual está implementado dentro dos nós "/G3T1_4" e "/G3T1_5";
  - Glucometro (glucosemeter_data): para medir os níveis de glicose no sangue. Esse sensor virtual está implementado dentro do nó "/G3T1_6".

- **Sensores físicos:** Estes, por sua vez, são usados para aferir o estado do paciente. Eles fazem parte do sistema como uma opção de entrada, ou seja, são opcionais. A versão da BSN estudada conta com dois sensores físicos disponíveis que fornecem dados para três sensores virtuais. Sua organização pode ser melhor observada na Figura 2.2. São esses sensores físicos:

  - Oxímetro de pulso e sensor de frequência cardíaca MAX30102: é um módulo pequeno e integrado usado para medir dois dados: a saturação de oxigênio no sangue e a frequência cardíaca. Ele utiliza um fotodetector e luzes LED vermelhas e infravermelhas em conjunto para rastrear mudanças na forma como os vasos sanguíneos, sob a pele, absorvem a luz. Para calcular a quantidade de hemoglobina oxigenada e desoxigenada presente, o sensor envia luz para o tecido e mede a luz que é refletida ou transmitida.

    O sensor pode determinar a frequência cardíaca e uma aproximação do nível de saturação de oxigênio monitorando a variação na absorção de luz. Para produzir leituras confiáveis, o sensor MAX30102 possui algoritmos de processamento de sinal que removem ruídos. Ele usa uma interface I2C para interagir com um microcontrolador ou um computador, simplificando a integração em uma variedade de aplicações, incluindo dispositivos vestíveis, rastreadores de condicionamento físico e aplicativos de saúde [18]. Ao usar esse sensor é importante se atentar a fontes externas de luz que podem interferir nas suas leituras.
  - Sensor de temperatura DS18B20: Este sensor resistivo tem como função aferir a temperatura, usando para isso, o princípio da termorresistência. Ele opera comparando a resistência de seu elemento sensor com um resistor de referência interno. A tensão analógica resultante é então digitalizada e transmitida

como um sinal digital ao microcontrolador, que pode então interpretar e exibir a leitura da temperatura. O sensor DS18B20 fornece medições precisas de temperatura e é amplamente utilizado em diversas aplicações, como monitoramento climático, controle industrial e automação residencial [19].

Na Figura 2.2 é possível observar o fluxo de dados a ser processado pelo próprio software. Outro ponto importante apresentado é a ligação do projeto com entidades médicas e de saúde externas, bem como o retorno ao próprio paciente. Nas Figuras 2.3 e 2.4 pode-se exemplificar o protótipo do hardware montado para teste. Ficam evidentes nessa montagem alguns componentes físicos que compõem o projeto. Na Figura 2.3 observa-se a placa Arduino Leonardo conectada aos sensores e ao Raspberry Pi, esse último sendo responsável por realizar a maior parte da execução dos processos. Também pode-se observar o sensor MAX30102 conectado pelo cabo *jumper* (colorido) na parte inferior da Figura, bem como a tela do código da BSN sendo executado. Já na Figura 2.4 pode-se observar o Raspberry Pi, responsável pela execução do software da BSN, e os sensores MAX30102 e DS18B20 interfaceados, cada um deles, por um Arduino Leonardo. Os sensores se conectam ao Arduino por meio dos cabos *jumper* e os Arduinos se conectam ao Raspberry por meio de cabos USB. Essa montagem está melhor descrita mais adiante.

2.3.3 Organização de software

Como parte central do projeto há o software da BSN, a parte vital que reúne os dados recolhidos pelos meios de entrada e os processa para obter um conjunto de previsões sobre o paciente. Este escopo pode ser dividido em quatro partes principais: Sistema de Gestão, Sistema Gerenciado, Repositório de Conhecimento e Módulo de Simulação. A Figura 2.5 representa a organização básica de cada um desses módulos. Nela temos cada componente representado por uma das caixas e as setas representando o principal fluxo de dados existente.

No que diz respeito a esses macro componentes e seus funcionamentos, uma breve descrição do papel de cada um é definido abaixo [1] e ilustrado na Figura 2.5:

- **Sistema de Gestão:** Este módulo é composto pelo Gerente de Estratégia e pelo Executor de Estratégia, sendo responsável pela implementação do controlador para lidar com o processo de adaptação. Este Gerenciador estima marcos de confiabilidade, reduzindo a margem de erro e tratando de inconsistências e custos, principalmente elétricos, dos componentes do módulo do Sistema Gerenciado. O Executor de Estratégia é onde o controlador é implementado, ele aplica estratégias de adaptação para atingir os pontos previamente estimados para cada componente. A Figura 2.5 mostra a interação entre esses componentes integradores do Sistema de Gestão e

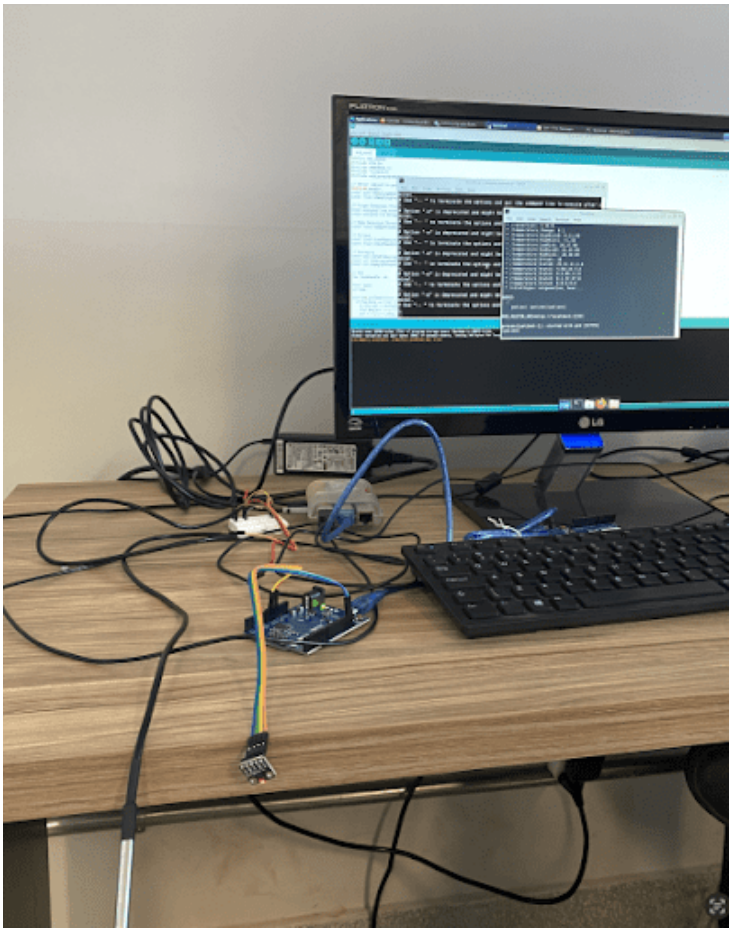


Figura 2.3: *Body Sensor Network* montada com sensores físicos usando uma placa Arduino Leonardo e uma placa Raspberry Pi.

as demais partes, bem como o ciclo fechado de *feedback* para o Executor de Estratégia, que funciona tanto para os marcos dos componentes do Sistema Gerenciado quanto para o sistema como um todo. Esse componente utiliza os pontos de ajuste estimados pelo Gestor de Estratégia e os contrasta com o valor real do atributo de qualidade desejado para cada componente, aplicando a política de adaptação. Neste caso, a política de adaptação consiste em ajustar as frequências dos componentes de acordo com o erro calculado através do controlador, de forma a simular a taxa de amostragem dos sensores e a taxa de processamento do *Central Hub*. Esses ajustes impactam diretamente na confiabilidade geral do sistema, pois espera-se que mais pontos de dados coletados por um sensor, por unidade de tempo, forneçam uma medição mais precisa do estado do paciente. Mensagens perdidas no *Central Hub* devido a problemas de fila, podem afetar a confiabilidade do sistema. Além disso, escolhas de frequência de medidas dos sensores impactam no consumo de energia dos componentes, já que mais execuções realizadas por unidade de tempo acarretam

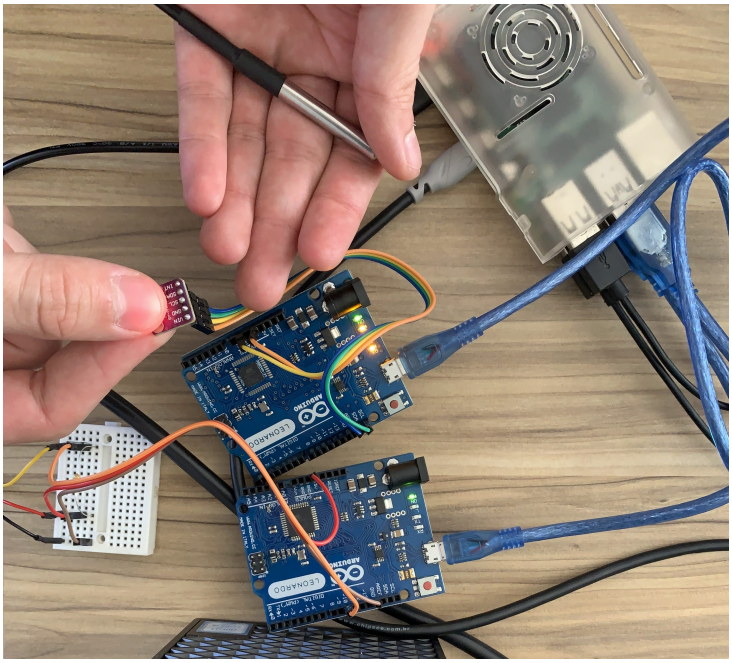


Figura 2.4: Sensores MAX30102 e DS18B20 usados junto a *Body Sensor Network* para coleta de dados reais.

em um maior consumo energético.

- **Sistema Gerenciado:** Este ponto compreende o *Central Hub* e os sensores de monitoramento dos sinais vitais que estão conectados a ele. Existem dois componentes responsáveis pela comunicação entre este módulo e os demais módulos da BSN. A comunicação entre esse módulo e os demais ocorre por meio de duas estruturas, as sondas e os efetores. As sondas são responsáveis por coletar dados dos componentes do Sistema Gerenciado e enviá-los tanto para o Repositório de Conhecimento quanto para o Sistema de Gestão. Os efetores são responsáveis por receber comandos de adaptação do Executor de Estratégia e alterar os parâmetros dos componentes do Sistema Gerenciado.
- **Repositório de Conhecimento:** Compreende as fórmulas paramétricas para cálculo da confiabilidade e consumo de energia do Sistema Gerenciado, os objetivos a serem alcançados e os registros do sistema. Os registros incluem Adaptações, *Status*, Eventos, Incertezas e *Status* de Energia, que armazenam informações relacionadas a comandos de adaptação, *status* de componentes, ativação/desativação, incerteza injetada e informações de custo. Este conjunto de dados é constantemente alimentado e analisado.
- **Módulo de Simulação:** Este é o componente de Injeção de Incertezas usado para

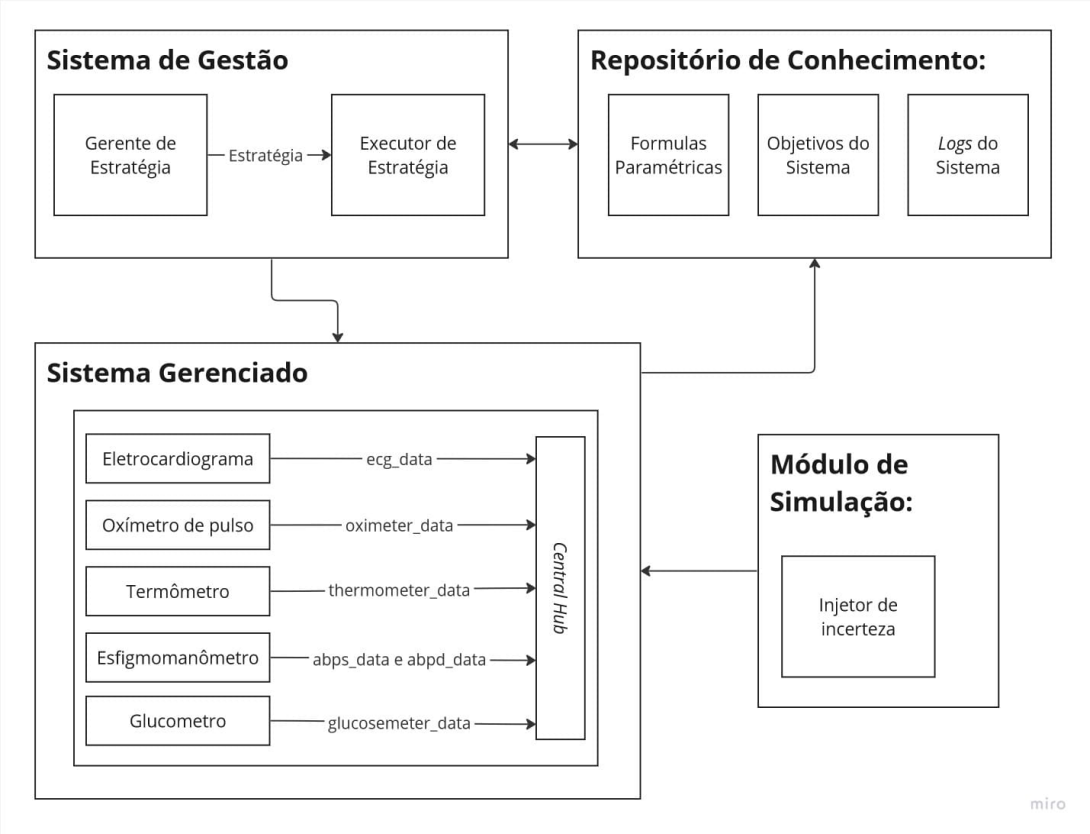


Figura 2.5: Perspectiva arquitetônica do BSN, e cada um dos seus principais módulos, como um exemplar autoadaptativo. Figura embasada no trabalho disponível em [1].

simular as incertezas previstas para o Sistema Gerenciado, podendo esse módulo ser ativado ou desativado dependendo da aplicação desejada. Este componente é responsável por injetar incerteza nos dados do sensor do Sistema Gerenciado para induzir falhas no processo de coleta de dados.

Todos estes módulos que compõem a BSN são coordenados pelas estruturas do *Robot Operating System* (ROS) através da troca de mensagens numa arquitetura do tipo *publisher and subscriber*, ou, em português, assinatura e publicação. Na Figura 2.6 estão representados os nós da *Body Sensor Network* e suas relações.

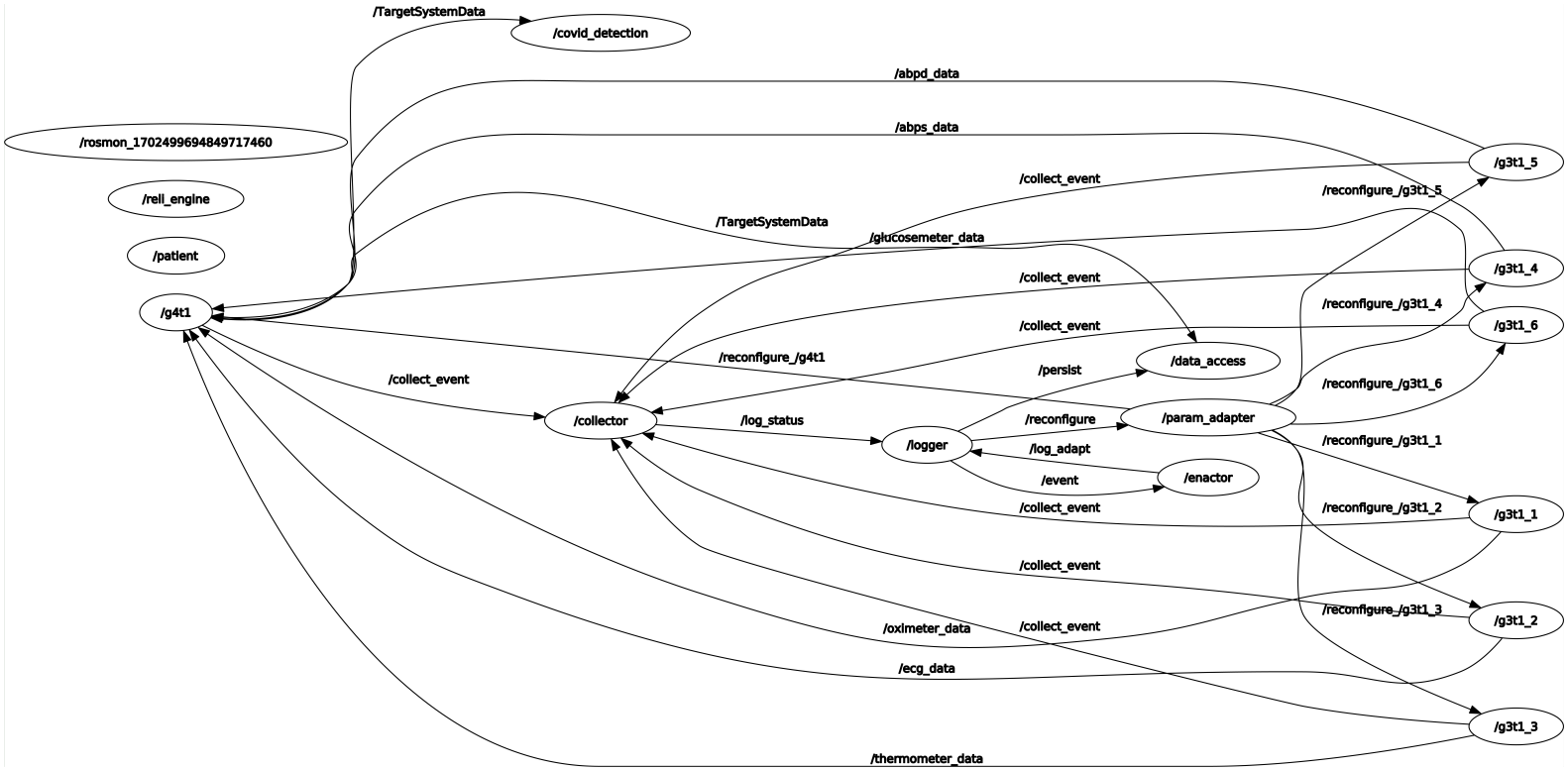


Figura 2.6: Nós e suas relações durante a execução da *Body Sensor Network*. As elipses representam cada um dos nós e as setas, os fluxos básicos de dados..

Na Figura 2.6 podem ser identificados alguns pontos interessantes. Nela tem-se uma elucidação dos nós da *Body Sensor Network*, representados pelas elipses, e as relações de fluxo de dados, representadas pelas setas. Observa-se na parte direita os seis sensores virtuais, nomeados de "/g3t1_1" a "/g3t1_6", sendo o mapeamento entre cada sensor virtual observável na lista 2.3.2, todos eles conectados ao nó "/collector" e ao nó "/g4t1". Esse segundo nó, o "/g4t1" é o responsável por reunir os dados do paciente, independente de qual seja a origem selecionada. Esses dados são usados para aferir o estado do paciente, calculando a faixa de risco para cada sensor e para a situação geral, e para identificar possíveis casos de covid-19. Essa identificação de casos suspeitos de covid-19 é feita no nó "/covid_detection", que recebe o conjunto de dados "/targetSystemData" diretamente do "/g4t1", tendo assim acesso aos dados de todos os sensores.

2.4 Sistemas Autoadaptativos

Alguns sistemas de software modernos podem operar em ambientes incertos e devem adaptar-se continuamente. Para que esses sistemas possam executar tarefas complexas e autônomas em ambientes sujeitos a eventos imprevisíveis e, ainda assim, obter os resultados próximos aos esperados, é fundamental que o modelo de arquitetura de controle de robô seja autoadaptativo [20]. Essas incertezas podem surgir devido a mudanças no ambiente operacional, variações na disponibilidade de recursos e mudanças nas metas do usuário. Geralmente, os operadores do sistema são responsáveis por lidar com estas incertezas, mas esta tarefa pode ser complexa, propensa a erros e dispendiosa. O conceito de autoadaptação visa permitir que o sistema colete dados adicionais durante sua operação para se gerenciar com base em objetivos de alto nível. Esses dados são utilizados para resolver incertezas e, com base em seus objetivos, configurar ou ajustar o sistema para atender às condições enfrentadas [21].

Um sistema auto adaptativo é capaz de ajustar seu comportamento em resposta a mudanças no ambiente e no próprio sistema. O prefixo "auto" indica que o sistema toma decisões de forma autônoma, ou com intervenção humana mínima, para se adaptar às mudanças no seu contexto e ambiente. Portanto, entende-se inicialmente um sistema autoadaptativo como uma caixa preta, onde a autoadaptação é uma propriedade que permite lidar com mudanças nas condições externas, como disponibilidade de recursos, demandas de trabalho e ocorrência de falhas e ameaças, enquanto a caixa continua a fazer o seu trabalho, ou seja, um sistema autoadaptativo é capaz de captar as alterações nas características do ambiente e no objetivo e se adaptar a nova realidade de modo a garantir a integridade do sistema em tempo real [20].

O estudo de sistemas autoadaptativos é importante para esse projeto uma vez que a versão base da BSN adotada é um sistema autoadaptativo. Também denominada como *Self-Adaptive - Body Sensor Network* (SA-BSN), ela conta com uma capacidade de se adaptar a realidade do ambiente que a cerca e reestruturar alguns de seus comportamentos para isso. Um exemplo para isso é a consciência no consumo de recursos energéticos, como no caso da bateria, relacionando a frequência de leitura dos sensores ao recurso disponível.

2.5 Sistemas Embarcados

Sistemas embarcados são dispositivos eletrônicos capazes de processar dados integrados a outros produtos ou dispositivos, para executar funções específicas [22]. Eles são projetados para atender a requisitos específicos de desempenho, confiabilidade e eficiência energética. Alguns exemplos comuns de sistemas embarcados incluem dispositivos de automação residencial, sistemas de navegação de veículos, eletrodomésticos inteligentes, *drones*, entre outros. Esses sistemas estão presentes em diversos aspectos do cotidiano, proporcionando maior comodidade, eficiência e automação em diversos setores.[22]

Nesse projeto, esses sistemas são parte primordial do hardware utilizado para execução do projeto, manifestando-se nas figuras do Arduino e do Raspberry Pi. Tais tecnologias encontram-se discorridas de forma mais aprofundadas abaixo.

2.5.1 Arduino

Para interligar o sistema computacional da *Body Sensor Network* e os sensores físicos utilizou-se uma placa Arduino Leonardo. Arduino é uma plataforma eletrônica de código aberto que consiste em componentes de hardware e software. Ele fornece uma maneira fácil de criar projetos interativos [23]. Essas placas são equipadas com microcontroladores que podem ser programados para controlar diversas entradas e saídas, permitindo aos usuários construir uma ampla gama de dispositivos eletrônicos, ou seja, uma rede de sensores [24].

O Arduino Leonardo, representado na Figura 2.7 em especial, é uma placa que se destaca pelas suas características únicas. É baseado em um microcontrolador que possui recursos de comunicação USB integrados. Isso significa que o Leonardo pode aparecer como um teclado ou mouse para um computador, permitindo emular pressionamentos de teclas ou movimentos do mouse. Esse recurso é particularmente útil para projetos que envolvem interação humano-computador, como a criação de dispositivos de entrada personalizados ou a automação de tarefas. Além disso, o Leonardo possui um número maior de pinos digitais e analógicos em comparação com algumas outras placas Arduino,

proporcionando maior flexibilidade na conexão de sensores, atuadores e outros componentes eletrônicos [25]. No Arduino da Figura 2.7, pode ser observado no canto superior esquerdo a sua porta micro USB, usada para realizar a conexão serial entre essa placa e a unidade de processamento central escolhida, unidade que pode ser um computador ou um micro computador, por exemplo. Mais ao centro existe um quadrado preto, que é seu micro controlador, e nas partes superiores e inferiores, suas portas disponíveis para conexões com sensores e atuadores. É esse conjunto de portas que se conecta os sensores físicos a serem usados.

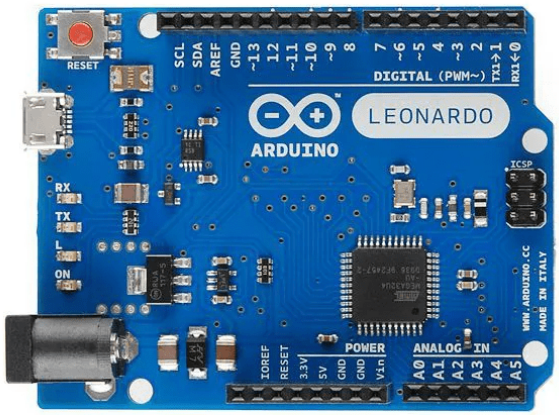


Figura 2.7: Placa Arduino Leonardo vista de cima. Fonte: MakerHero.

2.5.2 Raspberry Pi

O Raspberry Pi é um computador de placa única pequeno e acessível que ganhou imensa popularidade por sua versatilidade e acessibilidade. Foi desenvolvido com o intuito de promover o ensino da ciência da computação e capacitar os indivíduos a mexer com eletrônica e programação. O Raspberry Pi é equipado com processador, memória, portas de entrada/saída e executa um sistema operacional, denominado atualmente Raspberry Pi OS, anteriormente chamado de Raspbian, que é baseado em Linux. Ele pode ser conectado a um monitor, teclado e mouse, ou a outros periféricos, funcionando como um computador básico.

Um dos pontos que diferencia o Raspberry Pi, além de seu custo reduzido, é sua capacidade de interagir com o mundo físico por meio de seus pinos *General Purpose Input/Output* (GPIO). Esses pinos permitem de maneira facilitada a conexão de sensores, atuadores

e outros componentes eletrônicos, pelos usuários, tornando-os uma excelente plataforma para a criação de projetos relacionados à automação residencial, robótica,*Internet* das Coisas (IoT) e muito mais. O baixo custo, o tamanho compacto e o amplo suporte da comunidade do Raspberry Pi tornaram-o uma escolha popular para amadores, educadores e profissionais [26].

Ao observar a Figura 2.8 se encontra na parte superior esquerda os pinos de *General Purpose Input/Output* (GPIO) e na parte mais a direita um conjunto de portas USB, que podem ser usadas para estabelecer uma conexão serial entre essa placa e uma placa Arduino.

Dentro desse projeto o Raspberry Pi é utilizado como opção de hardware para execução da *Body Sensor Network*, podendo o sistema principal ser operado em um computador portátil, um computador fixo ou um mini computador. Uma imagem do sistema montado usando tanto a placa Arduino quanto a placa Raspberry Pi pode ser observada na Figura 2.3.

Figura 2.8: Placa Raspberry Pi 3 Modelo B e sua caixa de revenda. Fonte: MSato.

2.6 *Suport Vector Machine* (SVM)

As *Suport Vector Machine*, ou, em português Máquinas de Suporte Vetorial, constituem uma técnica de aprendizado de máquina que vem recebendo crescente atenção da comunidade de *Machine Learning* (AM) e muitas vezes possui resultados comparáveis a de Redes Neurais Artificiais (RNAs) [27]. Essa técnica tem sucesso em áreas como ca-

tegorização de textos, na análise de imagens e em Bioinformática [27]. Dentro do ramo da aprendizagem de máquina, SVM é um modelo de aprendizado supervisionado. Ele é usado com um conjunto de dados rotulado, como no caso de pacientes que podem estar infectados com covid-19 ou não. Nesse modelo se coloca dados a serem analisados, posteriormente ao seu treinamento, para predição em qual dos grupos esse dado se encaixa [28].

Existem algumas funções e tipos de *Suport Vector Machine* (SVM), abaixo estão listadas brevemente, a título de referencia, algumas das principais. Nesse trabalho o foco será na classificação binária, podendo ser usada para classificar entre casos positivos e negativos para a suspeita de covid-19.

- Classificador Binário Linear: Este é o caso mais simples de SVM, onde os dados são linearmente separáveis. Nele o treinamento busca o hiperplano que maximiza a margem entre as classes, onde a margem é definida como a distância perpendicular entre a fronteira de decisão e os dados mais próximos a ela [29].
- Classificador Não-linear: Enfrentando cenários em que os dados não são linearmente separáveis, o SVM transforma os dados em um espaço onde eles se tornam linearmente separáveis, aumentando a dimensão do espaço trabalhado. Isso permite que o modelo lide com problemas de classificação mais complexos [30].
- SVM Multiclasse: O *Suport Vector Machine* (SVM) é originalmente projetados para classificação binária, ou seja, separar em apenas dois grupos. No entanto, essa técnica pode ser estendidos para lidar com problemas de classificação do tipo multiclasse, ou seja, mais de duas classes de classificação. Uma abordagem comum é a estratégia "*one-vs-all*", onde um classificador SVM é treinado para cada classe contra todas as outras classes, realizando uma combinação de classificações binárias [31].
- Regressão por SVM: A regressão por SVM, também conhecida como *Support Vector Regression* (SVR), é uma extensão do SVM que permite a previsão de valores contínuos [32].

De forma geral, no problema de classificação binária, os dados dos dois grupos são distribuídos em um campo, em seguida é marcado um vetor, como uma reta ou um plano, que delimite uma divisão entre esses dois grupos, com a maior margem possível de distância deles. Esse caso pode ser melhor observado na Figura 2.9, na qual existe uma representação bidimensional de um *Suport Vector Machine* (SVM). Nela há dois grupos distintos, representados por circunferências preenchidas e vazias, e três linhas que dividem esses dois grupos. Podemos constatar cada uma das linhas e reparar que:

- H_1: não separa as classes.
- H_2 Separa as classes, mas apenas com uma pequena margem.
- H_3 os separa com a margem máxima, sendo o melhor para o esse caso da pesquisa em questão.

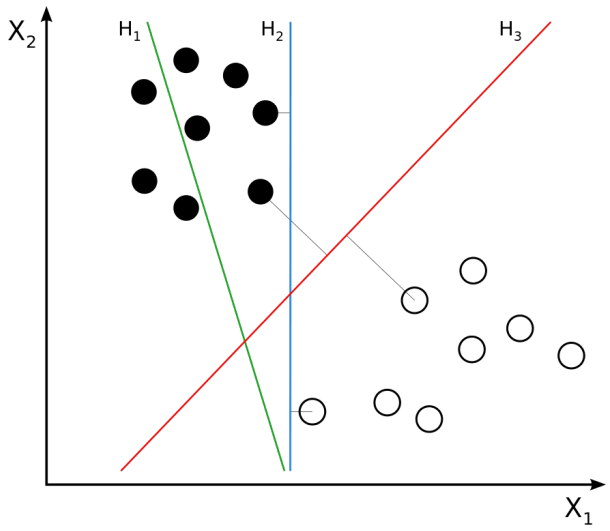


Figura 2.9: Gráfico mostrando como uma máquina de vetores de suporte escolheria um hiperplano de separação para duas classes de pontos em 2D. Fonte: Wikimedia Commons.

2.7 *Naïve Bayes* (NB)

Naïve Bayes (NB) é um classificador simples e popular sendo uma poderosa técnica de *Machine Learning* (AM). Foi verificado como o classificador probabilístico altamente profissional que possui sólidos fundamentos matemáticos [33], ele conta com fortes suposições de independência entre as variáveis sendo um dos modelos mais simples de redes bayesianas. Esse algoritmo funciona muito bem em vários casos complexos de aplicações no mundo real, casos como diagnósticos médico, predições em tempo real, filtragem de spam e previsão do tempo, apesar de sua suposições simplificadas e seu design ingênuo (*Naïve*) [33].

2.8 Matriz de confusão

A matriz de confusão, ou, também conhecida em alguns casos, de matriz de erro ou de tabela de confusão, é um termo usado na área de *Machine Learning* (AM) e representa

uma tabela útil para a visualização do desempenho de um algoritmo de classificação. Nesse instrumento, cada linha da matriz representa instâncias de uma classe prevista enquanto cada coluna representa instâncias da classe real, ou seja, compara os resultados esperados com os obtidos. Nesse trabalho, essa técinica será usada principalmente junto aos modelos da *Suport Vector Machine* (SVM) para analisar os resultados obtidos. Na Tabela 2.1, pode-se observar os campos esperados em uma matriz de confusão e na Figura 4.1 pode-se observar um exemplo dessa matriz.

		Casos reais	
		Condição positiva	Condição negativa
Casos previstos	Condição positiva prevista	Verdadeiro positivo	Falso positivo
	Condição negativa prevista	Falso negativo	Verdadeiro negativo

Tabela 2.1: Tabela representando uma Matriz de confusão e quais são seus campos esperados

Capítulo 3

Arquitetura e Implementação

Neste capítulo serão abordados os principais aprimoramentos e refatorações feitas na *Body Sensor Network* (BSN) para a criação do modelo de detecção de covid-19 e sua implementação dentro da BSN. De forma mais ampla, apresentamos de forma esquemática na Figura 3.1 o fluxo seguido em nossa abordagem para identificar casos suspeitos de covid-19 na BSN. Tal fluxo consiste inicialmente na entrada de dados simulados, externos e reais, formando a etapa de "Coleta de dados" (Seção 3.2). Seguido pela análise (Seção 3.3), na qual a base de dados fornece informações de teste e de treinamento para o Modelo SVM. Por fim, a interface do usuário é responsável pela a apresentação do resultado (Seção 3.4). Antes de apresentar as seções com uma visão detalhada dessas partes que compõem o fluxo do processo de identificação dos casos suspeitos de covid-19 na BSN, na seção a seguir apresentamos a visão arquitetural da BSN com foco nos novos módulos que a compõem.

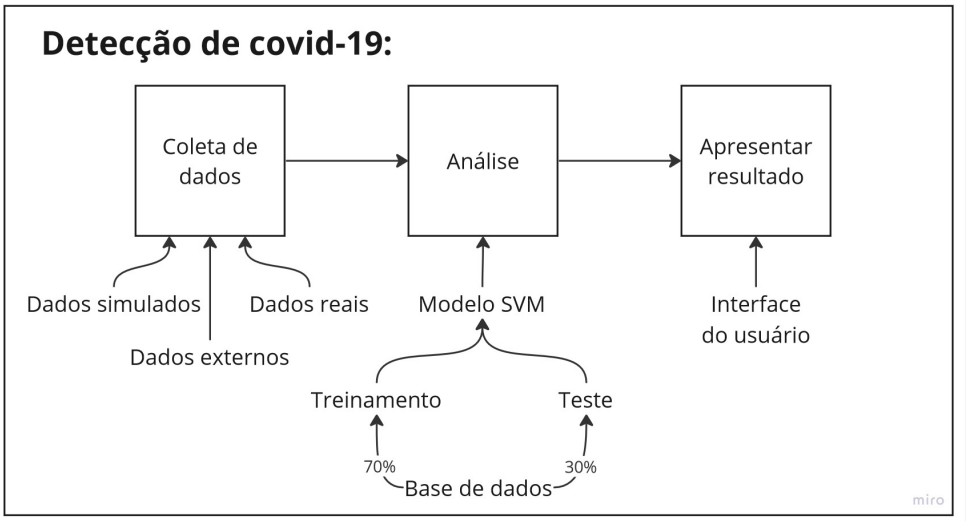


Figura 3.1: Fluxo do processo de identificação de casos suspeitos de covid-19, começando na aquisição dos dados e finalizando na apresentação do resultado.

3.1 Nova Arquitetura Proposta

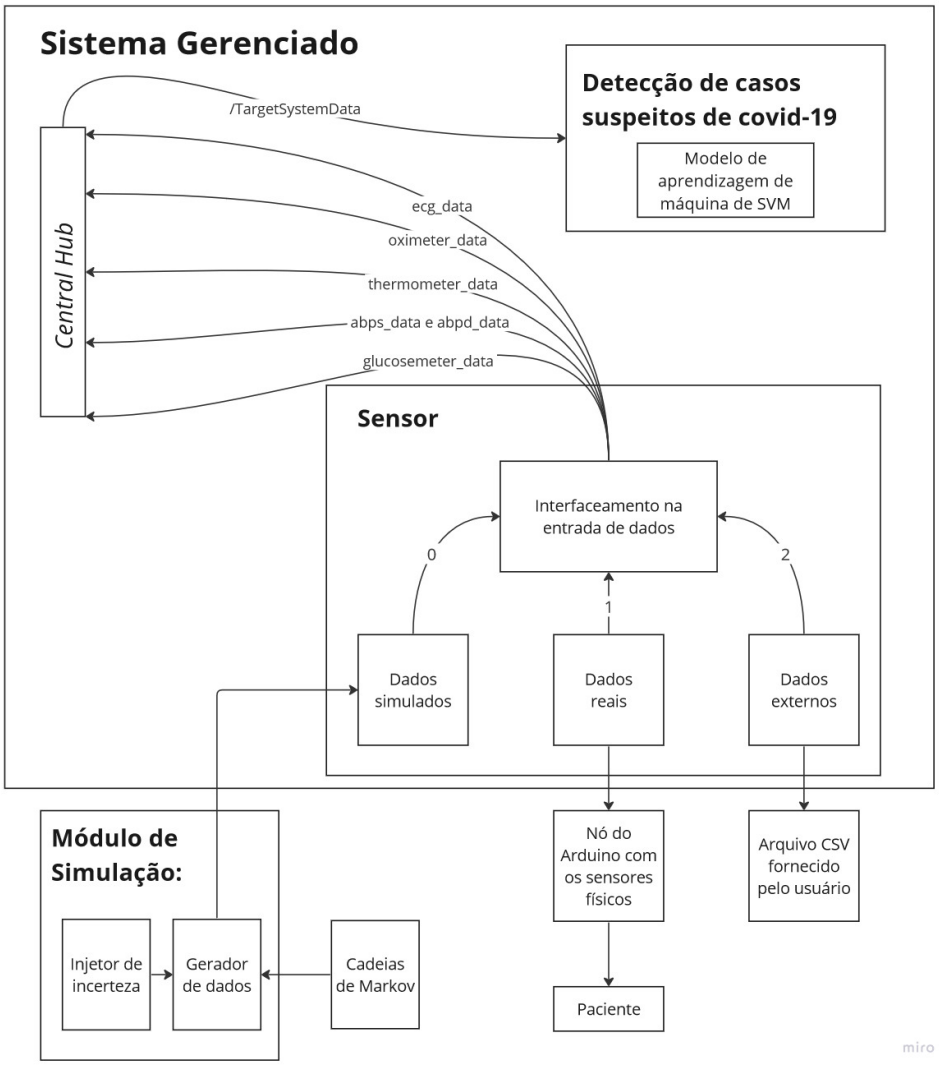


Figura 3.2: Representação gráfica da arquitetura da BSN implementada no projeto.

Uma visão da nova arquitetura estendida da BSN é apresentada na Figura 3.2, onde se pode observar as estruturas implementadas e suas relações. Explicamos de forma mais detalhada os módulos mais relevantes para o contexto deste trabalho: o Sistema Gerenciado e o Módulo de Simulação. O Sistema Gerenciado conta com alguns importantes elementos, são eles:

- ***Central Hub:*** Composto pelo nó ”g4t1”, é ele que orquestra o funcionamento geral do sistema. Ele recebe os dados dos sensores e calcula os riscos do paciente

- **Sensor:** Dentro desse ponto existem as fontes de dados que são interfaceadas, fazendo com que os dados sejam fornecidos de maneira independente da sua origem. São as origens adotadas:
  - Dados simulados: Esses dados são provenientes do Módulo de Simulação
  - Dados reais: Esses dados originam dos nós de dados gerados pelos sensores físicos. Os sensores físicos aferem o estado fisiológico do Paciente.
  - Dados externos: Esses dados são fornecidos pelo operador no formato de um arquivo CSV contendo os valores desejados.
- **Detecção de casos suspeitos de covid-19:** Esse módulo usa como base o Modelo de aprendizado de máquina de SVM para processar os dados do tópico "/TargetSystemData" e responder ao usuário se foi detectado um caso suspeito de doença por coronavírus ou não.

Já o Módulo de Simulação é responsável por gerar dados simulados, usando como base um conjunto de dados da Cadeia de Markov contendo faixas de riscos e probabilidades de transição.

3.2 Coleta de dados: Interfaces de entradas para detecção de covid-19

Um dos objetivos iniciais é a seleção de diferentes entradas para cada sensor da BSN. Para possibilitar essa seleção foi implementada uma seleção numérica, assim como apresentado na Figura 3.3 que traz um recorte do arquivo "g3t1_1.launch", usado como exemplo para todos os aquivos ".launch" dos sensores. Tais arquivos estão disponíveis na pasta "configurations > target_system".

```
<!--    Identifies where data entry should be made.
        0 -> Simulation (default)
        1 -> Real sensors with Arduino
        2 -> Table data (Turn on this feature to auto generate the file at data_to_read/*sensor_name*.csv)-->
<param name="connect_sensor_/g3t1_1" value="0" type="int" />
```

Figura 3.3: Trecho do código a ser usado pelo usuário para realizar a seleção da fonte de dados para cada sensor.

Na imagem observa-se uma variável do tipo inteiro (*int*) denominada "connect_sensor _/g3t1_1", onde g3t1_1 representa o sensor de oxigenação no sangue, que pode ser editada no arquivo "<nome do nó do sensor>.launch" da BSN. Aqui existem três possibilidades de entradas esperadas, cada uma descrevendo uma origem de dados para ser

usada pelo sistema. Essa arquitetura está apresentada na Figura 3.4, onde os dados a serem fornecidos ao *Central Hub*, a central de processamento, são interfaceados entre uma das três entradas disponíveis. São elas:

Figura 3.4: Fluxograma representando o interfaceamento de dados de entrada na BSN.

- Valor 0: Seleção padrão. Usa dados simulados gerados pelo próprio sistema;
- Valor 1: Usa dados reais coletados por sensores aferindo um paciente real, usando um Arduino Leonardo para realizar o interfaceamento entre os sensores e o sistema principal da BSN;
- Valor 2: Dados provenientes de uma tabela Valores Separados por Vírgula (CSV) preenchida ou importada pelo usuário.

O trecho de código que recebe esse dado e efetiva a seleção da fonte de dados está representado no *Listing* 3.1, estando esse código disponível para cada um dos sensores em que é possível qualquer uma das três entradas. O trecho no Listing 3.2 representa a mesma função para os sensores da BSN que não possuem a possibilidade de entrada advindas de sensores reais interfaceados pelo Arduino. Como próximos passos é possível a expansão do repertório de sensores físicos disponíveis no projeto, permitindo o mapeamento de todos os sensores virtuais com sensores físicos.

Listing 3.1: Código do sensor para seleção de entrada

```
// Method that collects data from the origin and returns a
   double variable that represents the measured/generated
   value
double G3T1_1::collect() {
    double m_data = -1;

    // Switch that identifies where data entry should
       originate
    // 0 -> Simulation (default)
    // 1 -> Real sensors with Arduino
    // 2 -> Table data
    switch(connected_sensor){
        case 1:{
            m_data = collect_real_sensor();
            break;
        }
        case 2:{
            m_data = collect_table();
            break;
        }
        default:{
            m_data = collect_simulation();
            break;
        }
    }


    battery.consume(BATT_UNIT);
    cost += BATT_UNIT;

    collected_risk = sensorConfig.evaluateNumber(m_data);

    return m_data;

}
```

Listing 3.2: Código do sensor para seleção de entrada sem sensor físico disponível

```
// Method that collects data from the origin and returns a
   double variable that represents the measured/generated
   value
double G3T1_5::collect() {
    double m_data = -1;

    // Switch that identifies where data entry should
       originate
    // 0 -> Simulation (default)
    // 1 -> Real sensors with Arduino - Not Implemented
    // 2 -> Table data
    switch(connected_sensor){
        case 1:{
            ROS_INFO("Function not implemented. Using
               simulation data");
            connected_sensor = 0;
            m_data = collect_simulation();
            break;
        }
        case 2:{
            m_data = collect_table();
            break;
        }
        default:{
            m_data = collect_simulation();
            break;
        }
    }

    battery.consume(BATT_UNIT);
    cost += BATT_UNIT;

    collected_risk = sensorConfig.evaluateNumber(m_data);

    return m_data;
}
```

Essa seleção pode ser feita para cada um dos sensores implementados disponíveis, o que permite misturar dados reais e simulados, por exemplo. Cada uma das opções de entrada de dados e suas implementações estão melhor definidas a seguir.

3.2.1 Coleta de dados - Dados simulados

Como origem de dados simulados tem-se um módulo da própria BSN que realiza essa geração de forma pseudoaleatória seguindo as regras de um conjunto de Cadeias de Markov. Esses dados oscilam dentro das transições estipuladas, seguindo um conjunto de probabilidades previamente definidas, e esse tipo de entrada pode ser usada para todos os sensores.

O caminho dos dados dentro desse sistema é o seguinte: O sensor, representado no arquivo "G3T1_1.cpp", por exemplo, requisita esse dado ao nó "/patient", no arquivo "PatientModule.cpp", que por sua vez solicita a geração efetiva dos dados no arquivo "DataGenerator.cpp". Essa geração usa como base os valores definidos na Cadeia de Markov, de faixas de risco e probabilidades de transições entre elas, que se encontram no arquivo "patient.launch" e podem ser modificados para criar outros perfis de pacientes.

No *Listing* 3.3 é possível ver como é feita a requisição desses dados e no *Listing* 3.4 pode-se ver a função que gera os dados solicitados.

Listing 3.3: Função collect_simulation que é usada para requisitar dados simulados gerados pela própria BSN

```
double Sensor::collect_simulation(){
    double m_data = -1;
    ros::ServiceClient client = handle.serviceClient<services
       ::PatientData>("getPatientData");
    services::PatientData srv;

    srv.request.vitalSign = name_node_sensor_simulation;

    if (client.call(srv)) {
        m_data = srv.response.data;
        ROS_INFO("new data collected: [%s]", std::to_string(
           m_data).c_str());
    } else {
        ROS_INFO("error collecting data");
    }
    return m_data;
```

```
}
```

Listing 3.4: Função calculateValue que é usada para gerar dados simulados requisitados

```
double DataGenerator::calculateValue() {
    if (markovChain.currentState > 4 || markovChain.
       currentState < 0){
        throw std::out_of_range("current state is out of
           bounds");
    }

    bsn::range::Range range = markovChain.states[markovChain.
       currentState];
    // Cria um numero aleatorio baseado no range
    std::uniform_real_distribution<double> value_generator(
       range.getLowerBound(), range.getUpperBound());
    double val = value_generator(seed);
    return val;
}
```

3.2.2 Coleta de dados - Dados externos

Como opção de entrada de dados, existem os dados externos, que podem ser usados por qualquer um dos sensores. Esses são dados que o usuário pode gerar por conta própria ou importar de algum banco de dados e salvar as leituras que deseja fornecer a *Body Sensor Network* (BSN) em um arquivo do tipo Valores Separados por Vírgula (CSV). Esse arquivo deve estar localizado na pasta ”data_to_read” e seguir o padrão de nome ”<nome do sensor>.csv”, como por exemplo ”bpm.csv”. Ao selecionar esse tipo de entrada, o arquivo para os dados é gerado automaticamente na primeira execução. Nesse arquivo cada leitura deve ser inserida em uma linha.

O caminho dos dados dentro desse sistema é o seguinte: O sensor, representado no arquivo ”G3T1_1.cpp”, por exemplo, tem seus dados de leitura requisitados pelo *Central Hub* (”g4t1”), e, com isso, o sensor usa o método que realiza a leitura do arquivo. Tal método pode ser observado no *Listing* 3.5

Listing 3.5: Função collect_table que é usada para recuperar os dados de um arquivo CSV

```
double Sensor::collect_table(){
    double m_data = -1;

    // Get the path to the sensor data file
    std::string path_file_to_read = get_current_dir_name();
    path_file_to_read += "/data_to_read/";
    path_file_to_read += name_node_sensor;
    path_file_to_read += ".csv";

    std::ifstream file;
    file.open(path_file_to_read);
    if(file.is_open()){
        // Search for the next line of the file
        std::string line;
        std::string csvItem;
        int line_number_counter = 0;
        bool last_line = 1;
        line_marker++;
        while (std::getline(file, line)) {
            line_number_counter++;
            if(line_number_counter == 1){
                m_data = std::stof(line);
            }
            if(line_number_counter == line_marker) {
                m_data = std::stof(line);
                std::cout << m_data << std::endl;
                last_line = 0;
                return m_data;
            }
        }
        if(last_line && m_data != -1){
            ROS_INFO("The end of the data file %s has been
               reached. Starting reading again", (
               name_node_sensor+".csv").c_str());
            line_marker = 0;
        }
```

```
        if(m_data == -1){
            ROS_INFO("Could not open the file or the file is
               empty. Using simulation data");
            connected_sensor = 0;
            m_data = collect_simulation();
        }
        file.close();
    }
    else{
        ROS_INFO("Could not open the file! Using simulation
           data");
        std::ofstream o(path_file_to_read.c_str());
        connected_sensor = 0;
        m_data = collect_simulation();
    }
    return m_data;
}
```

O uso desse método de entrada apresenta algumas peculiaridades:

- Caso o arquivo de dados não seja encontrado e esse tipo de entrada esteja selecionada, o arquivo será criado automaticamente.
- Quando o arquivo não é encontrado, está sem dados ou não pode ser lido, a entrada do sensor que estava definida como dados tabelados é alterada para dados simulados, garantindo assim, que o código possa continuar a ser executado.
- Quando todos os valores disponíveis no arquivo são recuperados e transmitidos ao sistema, é recomeçada a leitura do arquivo, sendo feito um *loop* nos dados disponibilizados pelo usuário.
- A BSN possui um filtro e um injetor de incerteza, ao usar dados externos tabelados, ambos são desativados para o sensor em questão.

3.2.3 Coleta de dados - Dados reais coletados por sensores físicos

Existe também a opção de coletar dados diretamente do paciente e encaminhá-los a BSN, usando para isso sensores físicos interfaceados com um ou mais Arduinos. No caso desse trabalho, foram usados dois sensores físicos, o MAX30102 (2.3.2) e o DS18B20 (2.3.2). O primeiro sensor é capaz de fornecer os dados de oxigenação no sangue e frequên-

cia cardíaca e o segundo a temperatura, ambos operando diretamente sobre o paciente. Na nota [1] pode-se baixar e utilizar o código dos sensores físicos junto ao Arduino.

Esse método de entrada, não está disponível para todos os sensores virtuais da BSN. Por limitações físicas dos sensores disponíveis, os sensores virtuais que podem ser usados nesse método para essa versão do software são os seguintes:

- Sensor de oxigenação no sangue: *oximeter* (/G3T1_1);
- Sensor de frequência cardíaca: ECG (/G3T1_2);
- Sensor de temperatura: *thermometer* (/G3T1_3).

No *Listing* 3.6 se observa o processo de solicitação de dados para o caso de sensores reais. Nesse caso, o dado do sensor é solicitado, fazendo o sensor uma requisição para um nó especifico, tal nó pode ser observado na Figura 3.5. Essa figura representa os nós e os tópicos da BSN quando ela é executada com o uso de sensores físicos. Nessa Figura, mais uma vez, as elipses representam os nós e as setas as relações entre eles por tópicos. É interessante observar que os sensores virtuais se encontram mais ao meio da figura e o "/g4t1", o *Central Hub*, está mais a direita. Na extrema direita está apresentado o nó "/covid_detection", implementado para executar o modelo de predição de casos suspeitos da doença por coronavírus. Na situação da Figura 3.5, esta sendo adotado dois sensores físicos mapeados para três sensores virtuais, gerando assim os dois nós observáveis no canto superior esquerdo. São eles os "/Arduino_oximeter" e o "/Arduino_termometer".

Listing 3.6: Função collect_real_sensor que é usada para recuperar os dados aferidos pelos sensores reais

```
double Sensor::collect_real_sensor(){
    double m_data = -1;

    std::string res;
    ros::ServiceClient client = handle.serviceClient<std_srvs
       ::SetBool>(name_node_sensor);
    std_srvs::SetBool srv;
    srv.request.data = true;
    if (client.call(srv)) {
        res = srv.response.message;
        m_data = std::stof(res);
```

[1] O código usado junto aos sensores no Arduino podem ser encontrados no link `https://github.com/carloseduardotl/bsn-sensors` [34].

```
        ROS_INFO("new data collected: [%s]", std::to_string(
            m_data).c_str());
    } else {
        ROS_INFO("error collecting data");
    }
    return m_data;
}
```

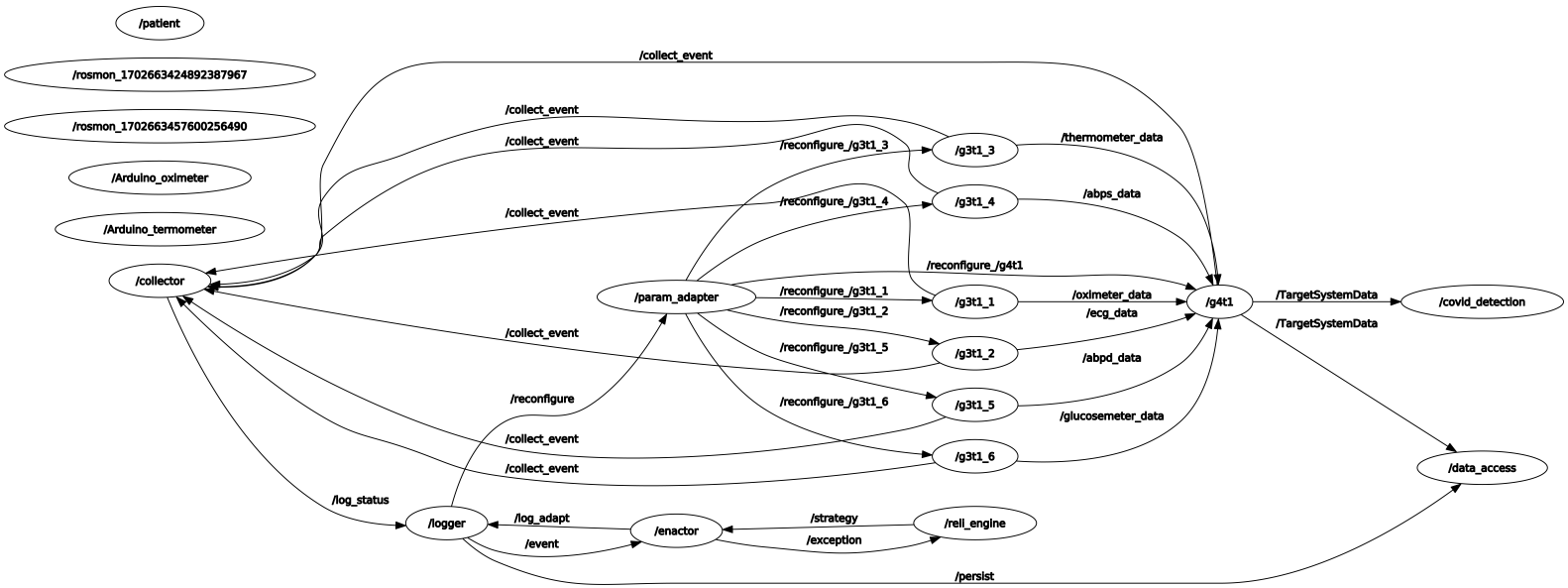


Figura 3.5: Nós e suas relações durante a execução da *Body Sensor Network* com sensores físicos. As elipses representam cada um dos nós e as setas os fluxos básicos de dados.

De forma geral, o uso de sensores reais pode parecer um pouco complexo, entretanto ele é composto de passos simples. Na Figura 3.6 os passos para o uso desse tipo de entrada de dados estão descritos, nela pode-se observar a divisão de três passos que orientam como deve ser feita a preparação para o uso de um ou mais sensores físicos. Fica ressaltado que alguns passos são lineares e outros não, ou seja, alguns precisam ser realizados previamente a outros e alguns não necessariamente.

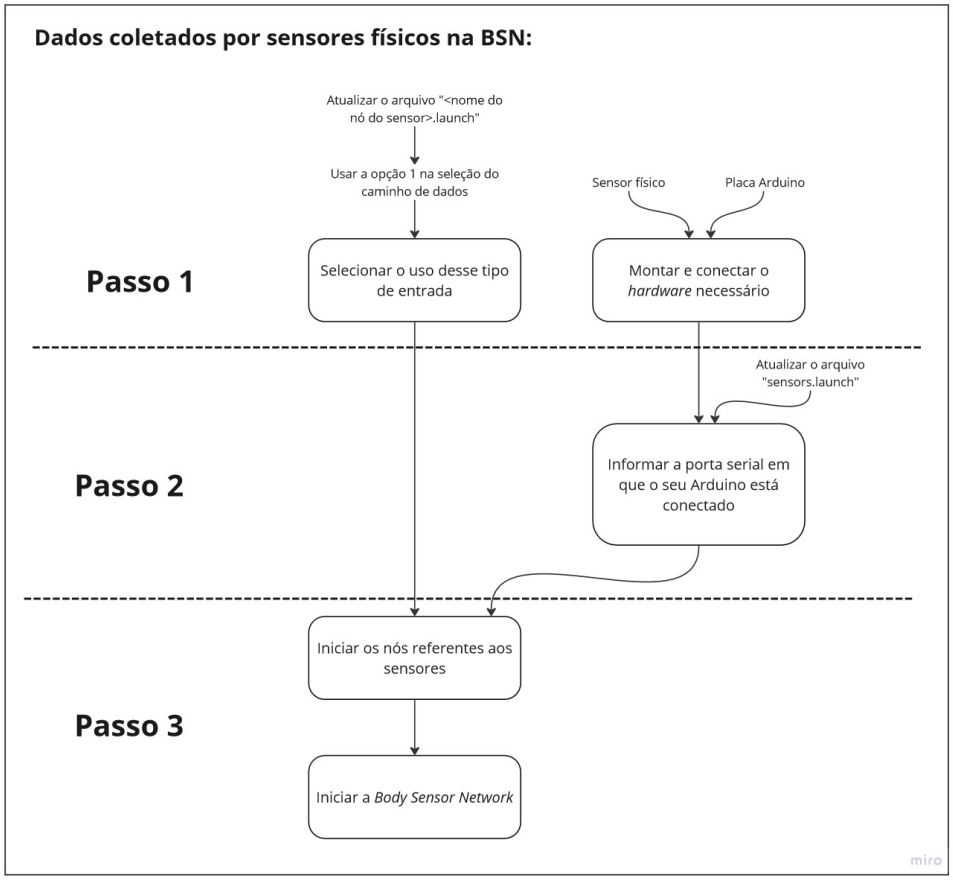


Figura 3.6: Indicações para o uso dos sensores físicos junto a BSN dividindo as ações necessárias em passos.

- **Passo 1:** Aqui se mostra essencial realizar duas ações primordiais, selecionar o tipo de entrada e preparar o hardware adicional necessário. A seleção do tipo de entrada deve ser feita de maneira semelhante às anteriores, editando a variável denominada "connect_sensor _/g3t1_<numero_do_sensor>", que se encontra no arquivo "<nome do nó do sensor>.launch" da BSN, agora sendo selecionada a opção 1. Além disso, se vê preciso realizar a conexão dos sensores fisicamente ao sistema que está executando a *Body Sensor Network* (BSN).

A montagem física pode ser observada na Figura 3.7, representado um esquema da conexão entre cada um dos dois sensores a um dos Arduinos Leonardo e a conexão dos Arduinos ao Raspberry Pi por meio de cabos USB. Nessa figura o Raspberry Pi, que atua como central de processamento para a BSN, está a esquerda e os dois Arduinos, que se conectam a ele, no centro. Mais na parte da direira estão representados os dois sensores, na parte superior o MAX30102 conectado diretamente ao Arduino por 4 cabos *jumper* e na parte inferior o DS18B20 conectado a uma *protoboard* com um resistor de $4,7K\Omega$. Na Figura 2.4 pode ser observado esse circuito montado com componentes reais e usado para aferir os dados do paciente junto a BSN.

- **Passo 2:** Esse passo consiste em informar o programa da BSN em qual porta serial do computador cada Arduino utilizado está conectado. Essa porta varia de computador para computador e é escolhida pelo Sistema Operacional (SO) da própria máquina, podendo ser o programa Arduino IDE [23] usado para identificar qual é a porta usada. Para visualizar esse dado, é preciso abrir esse ambiente de programação e selecionar a opção "*Tools*" > "*Port*", onde esse dado será apresentado. A Figura 3.7 traz o trecho de código do arquivo "sensors.launch" que cria os nós dos sensores físicos. Pode ser observado que na parte superior é criado o nó do sensor de temperatura e na parte inferior o nó do sensor de oxigenação e frequência cardíaca, bem como os parâmetros usados para isso. Essa Figura 3.7 apresenta onde a identificação da porta serial adotada deve ser inserida, na variável "*value*", ao lado da "*port*". A título de exemplo, o valor atualmente apresentado para o Arduino com o sensor de temperatura é "/dev/ttyACM0".

- **Passo 3:** Como último passo executa-se o programa, sendo importante executar primeiro o arquivo "sensors.launch", apresentado na Figura 3.7, e, posteriormente, executar a *Body Sensor Network* usando o arquivo escolhido, no caso adotou-se o "bsn.launch".

Listing 3.7: Código do arquivo "sensors.launch"que realiza a indicação das portas seriais e inicia os nós indicados

```
<launch>
  <node pkg="rosserial_python" type="serial_node.py" name="
     Arduino_termometer">
    <param name="port" value="/dev/ttyACM0"/>
  </node>
```

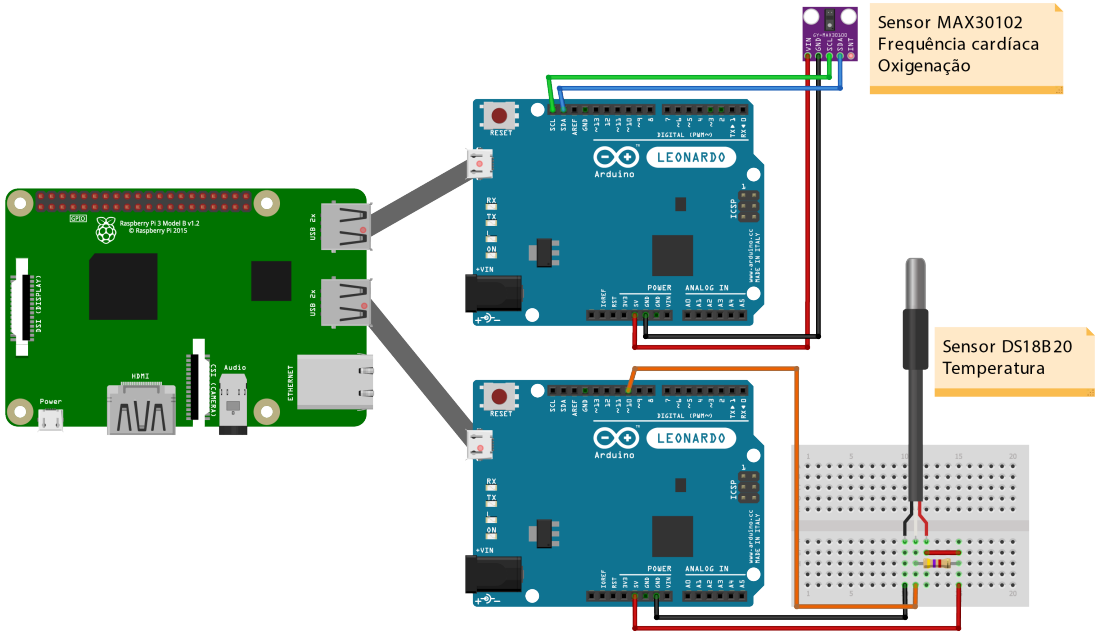


Figura 3.7: Esquema de montagem física dos sensores MAX30102, na parte superior da figura, e DS18B20, na parte inferior, ligados a Arduinos Leonardo que, por sua vez, estão ligados a um Raspberry Pi.

```
  <node pkg="rosserial_python" type="serial_node.py" name="
     Arduino_oximeter">
    <param name="port" value="/dev/ttyACM1"/>
  </node>
</launch>
```

3.3 Análise de dados - Modelo de SVM para detecção de casos suspeitos de covid-19

Para realizar a detecção de casos suspeitos de covid-19, usa-se o código disponível no Anexo I para treinar um modelo de aprendizagem de máquina de *Suport Vector Machine* (SVM) e exportá-lo. Esse modelo foi importado e utilizado na *Body Sensor Network* (BSN) para realizar predições dos casos suspeitos da doença com uma análise pontual dos sinais fisiológicos do paciente. Caso o quadro apresentado pelo paciente se aproxime ao definido pelo modelo como um quadro compatível com o de um indivíduo contaminado pelo covid-19, é indicado de que ele use o devido Equipamento de Proteção Individual (EPI), como uma máscara de proteção facial por exemplo, e procure suporte médico para

a devida aferição e comprovação da predição feita. Nesse projeto, foi escolhido um modelo do tipo *Suport Vector Machine* (SVM) pela sua facilidade de compreensão e utilização. Por ser um modelo simples e ligado diretamente a classificações binárias, ele se mostrou propicio para o desenvolvimento desse projeto.

3.3.1 Análise de dados - Base de dados usada para treinamento e teste do modelo

Para treinar e testar o modelo, foi adotado um conjunto de dados disponível aqui e abertos na plataforma Kaggle[2]. Esses dados foram descritos pelo autor como:

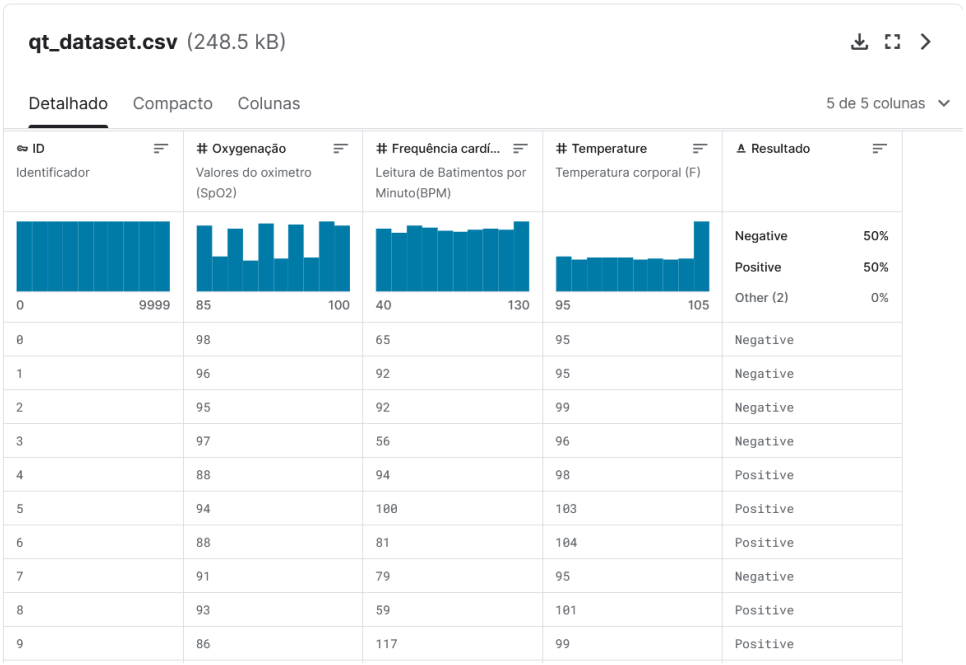
> *"Os dados são principalmente dados sintéticos e gerados artificialmente com base nas leituras iniciais do mundo real. Antes de gerar este conjunto de dados, foram registradas poucas leituras do mundo real, tanto pessoalmente quanto com ajuda online. Junto com isso, também tentamos encontrar uma faixa padrão de leituras vitais para as quais uma pessoa pode testar positivo para covid-19 (o que significa que a pessoa provavelmente está infectada com covid-19 se tiver os sinais vitais dentro da faixa padrão especificada). Usando esses pequenos dados reais existentes e um intervalo padrão, geramos esse conjunto de dados de 10.000 registros."*

Nesse conjunto de dados, encontrou-se dados semelhantes aos presentes na BSN, são eles:

- Identificador (Id): Número único usado para identificar o paciente;
- Oxigenação (*Oxygen*): Saturação de O_2 no sangue do paciente;
- Frequência cardíaca (*PulseRate*): Frequência cardíaca do paciente;
- Temperatura (*Temperature*): Temperatura em F^o do corpo do paciente;
- Resultado (*Result*): Situação do paciente para o diagnóstico de covid-19, podendo ser positivo ou negativo.

Um breve trecho dos dados pode ser visto na Figura 3.8, onde se observa um breve resumo gráfico dos dados contidos no arquivo, ao centro da imagem, de dados denominado "qt_dataset.csv" e um trecho de suas entradas, na parte inferior. Também existe um conjunto de gráficos de visualização do conjunto de dados de entrada na Figura 3.9, onde 0, em azul, representa casos negativos de covid-19 e 1, em laranja, representa os casos positivos. Nesse conjunto de gráficos está representada a separação inicial básica dos dados, ainda brutos, de diferentes maneiras visuais, sendo feita a correlação entre os casos positivos e negativos de covid-19 dentro de cada dupla de sensores.

[2]O banco de dados disponível na plataforma Kaggle está disponível no *link*: `https://www.kaggle.com/datasets/rishanmascarenhas/covid19-temperatureoxygenpulse-rate`



qt_dataset.csv (248.5 kB)

Detalhado | Compacto | Colunas | 5 de 5 colunas

ID Identificador	Oxygenação Valores do oxímetro (SpO2)	Frequência cardí... Leitura de Batimentos por Minuto(BPM)	Temperature Temperatura corporal (F)	Resultado
0 9999	85 100	40 130	95 105	Negative 50% Positive 50% Other (2) 0%
0	98	65	95	Negative
1	96	92	95	Negative
2	95	92	99	Negative
3	97	56	96	Negative
4	88	94	98	Positive
5	94	100	103	Positive
6	88	81	104	Positive
7	91	79	95	Negative
8	93	59	101	Positive
9	86	117	99	Positive

Figura 3.8: Trecho da planilha de dados usada para treinar e testar o modelo e breve resumo do conjunto de dados. Fonte: Captura de tela feita no site Kaggle traduzida.

No mesmo *link* dos dados é possível acessar alguns trabalhos de programação que se baseiam no mesmo banco de dados, possuindo dois projetos no momento de escrita desse documento, um incompleto e outro mais avançado. O trabalho desenvolvido por Nandalal D disponível na nota [3] merece uma atenção especial, nele o autor trabalhou em uma análise de dados parecida com a desejada nesse trabalho. Porém, ele desenvolveu um caminho diferente, e não o documentou em seu trabalho.

3.3.2 Análise de dados - Treinamento do modelo de SVM para detecção de casos suspeitos de covid-19

Para desenvolver o modelo de *Suport Vector Machine* (SVM) separa-se os dados iniciais em dois grupos, um de treinamento e um de teste. Para o grupo de treinamento 70% dos dados disponíveis são separados, totalizando 7.000 entradas, e para o teste 30%, contando com 3.000 entradas. Posteriormente, os dados provenientes da *Body Sensor Network* (BSN) são usados dentro do mesmo modelo para determinar o estado do paciente. A criação desse modelo e de algumas de suas estatísticas pode ser observado no Anexo I.

[3] Trabalho desenvolvido por Nandalal D: `https://www.kaggle.com/code/nandalald/predict-covid`

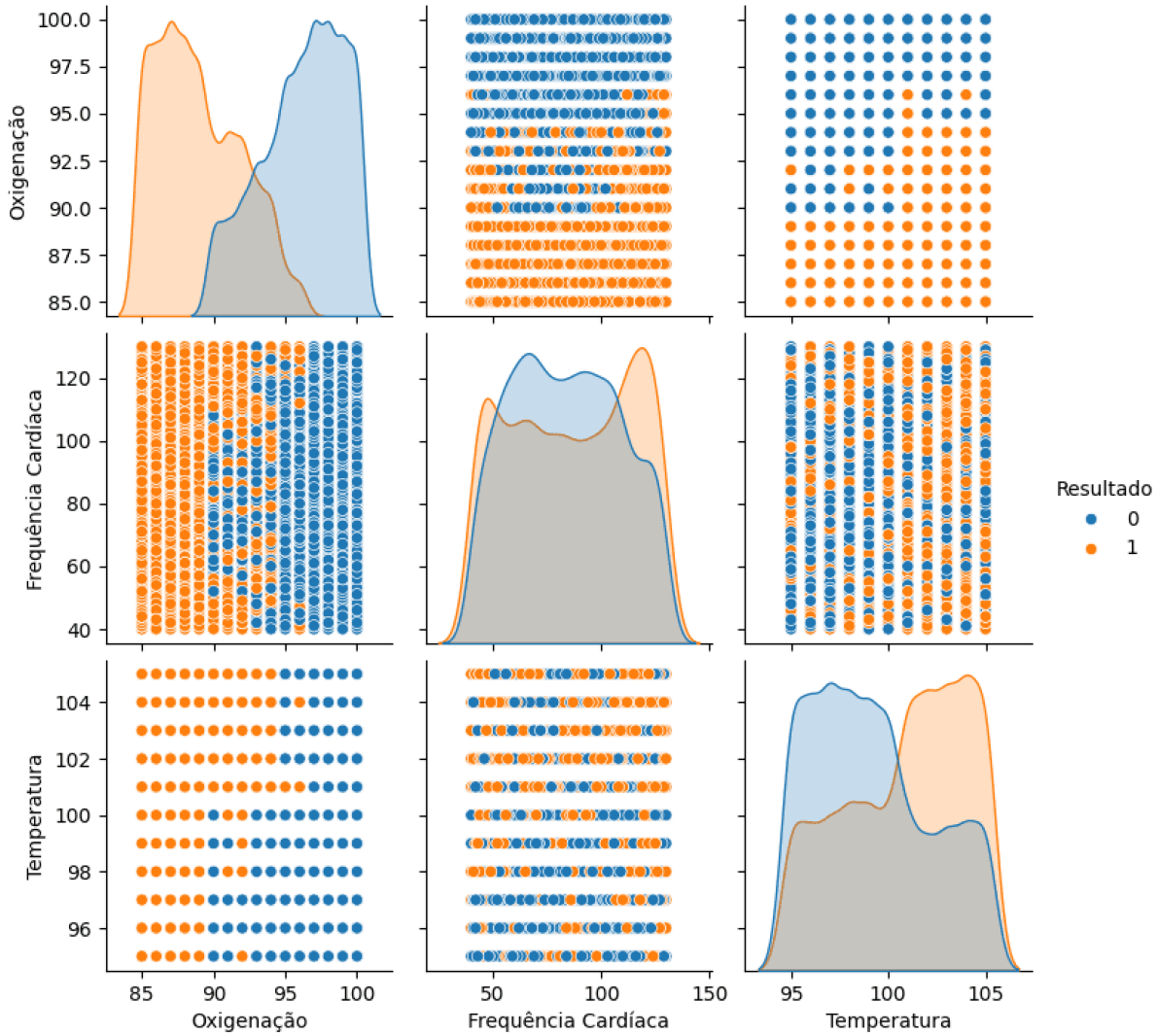


Figura 3.9: Conjunto de gráficos representando a entrada de dados bruta usada para treinamento e teste relacionados por sensor de origem. Na legenda do gráfico localizada a direita, o 0 representa os casos negativos para covid-19 e o 1 os casos positivos.

Bibliotecas adotadas

Um ponto importante para a execução do código é a adição de algumas bibliotecas em Python. Elas devem ser instaladas para que seja possível criar e usar o algoritmo de detecção de casos suspeitos da doença, ou seja, elas são essenciais para o uso do Anexo I e do Anexo III. Tais bibliotecas estão apresentadas abaixo, e é recomendado que o usuário possua a ferramenta "pip"[4] instalada junto ao seu Python, uma vez que ela auxilia na instalação e gerenciamento de pacotes dentro da linguagem em questão.

[4]A ferramenta "pip" está disponível na página `https://pypi.org/project/pip/`.

- **"pandas"[5]:** Essa é uma biblioteca que fornece estruturas de dados rápidas, flexíveis e expressivas, projetadas para tornar o trabalho com dados "relacionais" ou "rotulados" fácil e intuitivo. Ela é capaz de lidar com dados em diferentes formatos [35], sendo útil nesse caso para trabalhar com arquivos do tipo Valores Separados por Vírgula (CSV).
- **"sklearn"[6]:** É o módulo Python adotado para *Machine Learning*. Ele fornece ferramentas simples e eficientes para criar modelos de análise preditiva de dados [36], no caso atual uma *Suport Vector Machine* (SVM).
- **"matplotlib"[7]:** Essa biblioteca serve para criar visualizações estáticas, animadas e interativas em Python. Ela permite a criação de gráficos de alta qualidade e figuras interativas que podem ser ampliadas, deslocadas e atualizadas [37].
- **"pickle"[8]:** Esse é um módulo usado para serializar e desserializar uma estrutura de objeto Python. *Pickling* é o processo pelo qual uma hierarquia de objetos Python é convertida em um fluxo de *bytes*, e "*unpickling*" é a operação inversa, pelo qual um fluxo de *bytes* é convertido de volta em uma hierarquia de objetos [38]. De maneira mais específica, essa biblioteca se mostra útil no caso estudado ao ser usada para exportar e importar o modelo de detecção de casos suspeitos de covid-19.

De forma geral, nesse trabalho, adota-se; a biblioteca "sklearn", para ajudar na criação do modelo em si, e a biblioteca "matplotlib", para criar gráficos. Também é usada a biblioteca "pickle" para exportar o modelo criado, tirando a necessidade de uma nova criação do modelo, toda vez que for usada a BSN.

Criando o modelo

Os passos gerais realizados no código do Anexo I, responsável por gerar, exportar e testar o modelo de predição, são os seguintes:

1. Importar os dados brutos disponíveis e apresentar um gráfico resumindo-os, tal resumo encontra-se na Figura I.1 do anexo;
2. Separar o grupo de dados em dados de treinamento e de teste, compondo um conjunto com 70% do tamanho do original para treinamento e um com 30% para os de teste

[5]Página oficial da biblioteca "pandas": `https://pandas.pydata.org/`.
[6]Página com a documentação da biblioteca "sklearn": `https://pypi.org/project/scikit-learn/`.
[7]Página oficial da biblioteca "matplotlib": `https://matplotlib.org/`.
[8]Página com a documentação do "pickle": `https://docs.python.org/3/library/pickle.html`.

3. Separar os dados de oxigenação, Eletrocardiograma e temperatura dos resultados de positivo e negativo para covid-19 tanto no conjunto de treinamento quanto no de teste;

4. Treinar o modelo *Suport Vector Machine* com o grupo de dados de treinamento. Esse treinamento será melhor explicado na próxima seção;

5. Exportar o modelo criado para poder ser usado na BSN;

6. Testar o modelo criado com os dados de teste separados inicialmente e apresentar uma matriz de confusão reproduzida na Figura I.2 do anexo. Nessa figura pode-se observar no eixo horizontal os casos reais esperados e no eixo vertical os casos que foram previstos pelo modelo;

7. De forma análoga ao item anterior, aqui é feito o teste do modelo com os dados, porém dessa vez com os de treinamento, e apresentar uma matriz de confusão reproduzida na Figura I.3 do anexo. Nessa figura, também pode-se observar no eixo horizontal os casos reais esperados e no eixo vertical os casos que foram previstos pelo modelo

Explorando o modelo

Aqui é desenvolvida uma *Suport Vector Machine* (SVM) com o objetivo de encontrar o hiperplano em um espaço N-dimensional, onde N é o número de características adotadas, ou seja, 3 dimensões, que classifica distintamente os pontos de dados.

No Anexo I fica evidente uma linha usada para treinar o modelo, adotando o uso da biblioteca "sklearn", é ela

```
model = SVC(C=5,gamma="auto")
```

Essa linha cria uma instância da classe *Support Vector Classification* (SVC) da biblioteca "scikit-learn". SVC é a implementação de uma SVM para problemas de classificação e nessa classe estão sendo adotados dois parâmetros. Os parâmetros C e gamma são hiper parâmetros do modelo que influenciam o resultado final da classificação. Cada um deles é explicado:

- **C:** Este é o parâmetro é denominado de regularização ou de penalidade de erro. Ele determina o equilíbrio entre obter o menor erro de treinamento possível e maximizar a margem do hiperplano. Valores maiores de C resultam em uma menor margem do hiperplano e um menor erro de treinamento, ou seja, o modelo tenta classificar todos os exemplos de treinamento com mais precisão. No entanto, isso

pode levar a um sobreajuste (*overfitting*), onde o modelo se ajusta muito bem aos dados de treinamento, mas não generaliza bem para novos dados. Valores menores de C resultam em uma maior margem do hiperplano e permitem alguns erros de treinamento e isso pode levar a um modelo que generaliza melhor. No caso adotado, C é 5, o valor que melhor se adaptou segundo os testes efetivados. Sua definição foi feita por meio de testes empíricos, sendo escolhidos alguns parâmetros semelhantes e observados os resultados para o conjunto de dados de teste.

- **gamma:** Este é o parâmetro do *kernel*, no caso atual o *Radial Basis Function* (RBF), e controla a influência de cada exemplo de treinamento. Valores baixos significam que a influência de um exemplo de treinamento alcança longe, enquanto valores altos significam que a influência alcança pouco. Quando o gamma é definido como "auto", como no caso atual, ele será calculado como 1 dividido pelo número de características nos seus dados. Mais uma vez, esse valor foi testado junto aos dados separados. Para isso, e a função "auto" definida como padrão se mostrou a mais otimizada entre os casos verificados.

Modelo *Naïve Bayes* (NB)

Como comparação ao modelo *Suport Vector Machine* (SVM) foi testado um modelo *Naïve Bayes* (NB). Sua implementação seguiu a mesma que a apresentada no caso anterior e está apresentada no Anexo II, com a diferença da substituição do comando "model = SVC(C=5,gamma="auto")" pelo "model = GaussianNB(var_smoothing=1e-25)" e o uso da biblioteca "from sklearn.naive_bayes import GaussianNB". O comando está melhor decomposto abaixo.

```
model = GaussianNB(var_smoothing=1e-25)
```

Nele pode ser observado a criação do modelo do tipo *Naïve Bayes* (NB) que adotou apenas um parâmetro, o "var_smoothing" que é usado no classificador Gaussiano em questão, também do "sklearn", para melhorar a estabilidade dos cálculos. O NB assume que os dados de cada classe são distribuídos de acordo com uma distribuição normal, que pode levar a instabilidade numérica e, para evitar isso, o "var_smoothing" adiciona uma fração da variância máxima de todos os recursos aos quadrados das variâncias. Seu valor padrão é 1e-9. Isso significa que por padrão, adicionamos 1e-9 vezes a variância máxima a todas as variâncias. No caso atual foi adotado o valor 1e-25, assim como no caso do parâmetro "C" do modelo *Suport Vector Machine* (SVM), esse valor foi definido empiricamente por meio de testes de alguns valores e comparação de seus resultados para o grupo de dados de teste.

3.3.3 Análise dos dados - Implementação do modelo de detecção de covid-19 junto a BSN

No Anexo III é possível observar a implementação do nó na BSN responsável pela análise dos dados do paciente e retorno da previsão se ele apresentar um quadro semelhante a um paciente contaminado por covid-19 ou não. A relação desse nó pode ser observada na Figura 3.5, de forma geral ele recebe os dados da leitura do paciente do tópico "/TargetSystemData". De posse desses dados, ele usa o modelo previamente criado para enquadrar o cenário do paciente em um dos dois grupos, suspeito de contaminação por covid-19 ou não, e assim realizar um posicionamento. Os casos de respostas dele podem ser observados nas Figuras 3.10 e 3.11.

Nas figuras pode-se observar na parte da esquerda a identificação de cada nó de origem de cada mensagem, estando a mensagem em si apresentada na parte da direita. Na Figura 3.10 pode-se observar a mensagem do nó "/covid_detection" informando que "*[callback]: The patient's condition was compatible with a suspected case of Covid-19. Please see a doctor.*", ou seja, traduzindo, "[retorno de chamada]: O quadro do paciente era compatível com caso suspeito de Covid-19. Por favor, consulte um médico.", sugerindo que o paciente consulte um médico pela suspeita de um caso de covid-19 identificada no seu quadro. Já na Figura 3.11, a mensagem apresentada pelo mesmo nó é "*The patient's condition appears to be without signs of Covid-19. If you experience any symptoms, consult a doctor*", que traduzindo significa "[retorno de chamada]: O estado do paciente parece ser sem sinais de Covid-19. Se sentir algum sintoma, consulte um médico", informando que o quadro do paciente não foi identificado como suspeito de covid-19, mas que caso existam sintomas, um médico deve ser consultado.

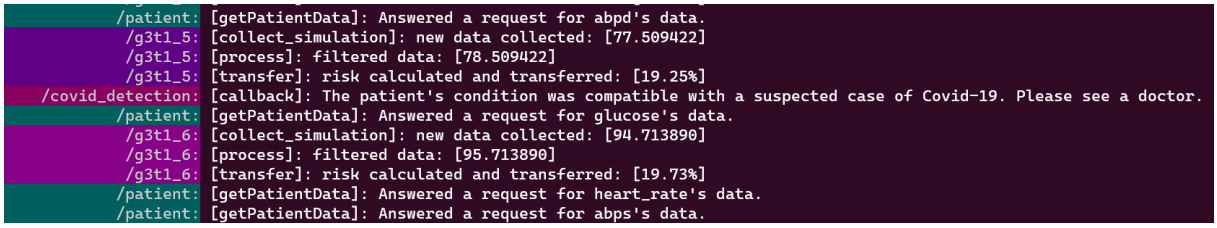


Figura 3.10: Resposta do nó covid_detection para casos classificados como suspeitos para covid-19.

O código usado e apresentado no Anexo III é dividido em duas funções, a "listener()" e a "callback(data)". A "listener()" inscreve esse nó no tópico "TargetSystemData" e executa a função "callback(data)" sempre que uma mensagem é publicada nesse tópico. Já a "callback(data)" realiza as seguintes ações:

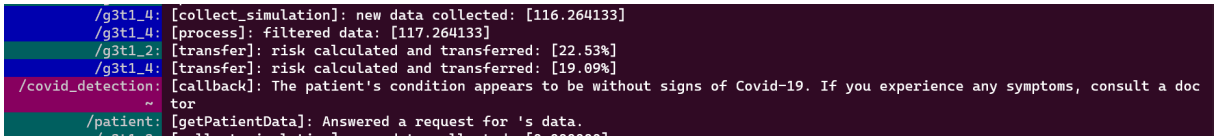


Figura 3.11: Resposta do nó covid_detection para casos classificados como não suspeitos para covid-19.

1. Importa o modelo de SVM criado pelo código do Anexo I usado para classificar o quadro dos pacientes como suspeitos de covid-19 ou não;
2. Cria um conjunto de dados para ser testado com base nos dados recebidos do tópico. Nesse conjunto de dados a temperatura de entrada é convertida, de graus Celsius (°C) para graus Fahrenheit (°F). Isso é crucial pois os dados de treinamento se encontram em Fahrenheit;
3. Os dados de entrada são avaliados pelo modelo importado, clasificando o conjunto de dados inserido;
4. Algumas informações podem ser apresentadas ao usuário para ajudar a acompanhar os dados usados para a análise;
5. O estado avaliado é então apresentado ao usuário.

3.4 Apresentação de resultados

Ao executar a *Body Sensor Network* (BSN) em um terminal, esse se torna a saída de dados padrão para o usuário e/ou operador. Ao longo desse trabalho repete-se telas como a apresentada na Figura 3.12. Ao se observar a tela, podem ser feitas algumas observações. Cada mensagem é composta por duas partes: a primeira colorida a esquerda descreve de qual nó da BSN se originou a mensagem, como o "/g4t1", por exemplo. Mais a direita existe a mensagem em si, que transmite as informações desejadas. Já na parte inferior observa-se a interface de comandos do usuário, onde se pode, por exemplo, escolher qual nó se deseja ignorar ou mostrar.

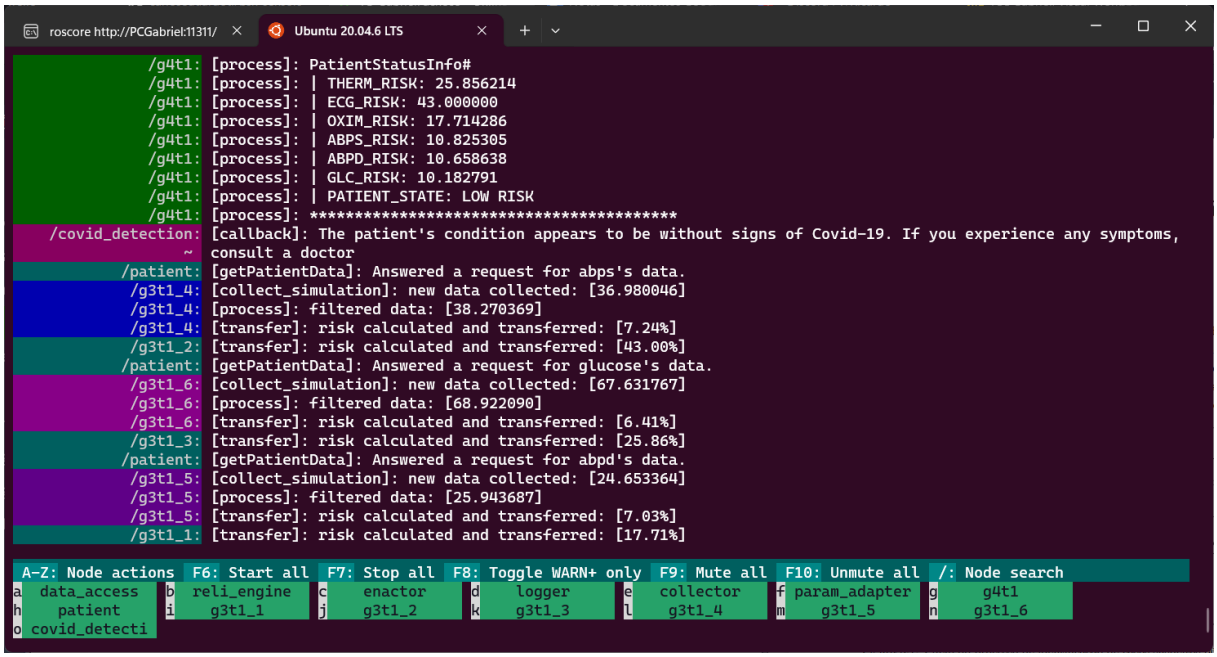


Figura 3.12: Exemplo de tela da BSN por meio de captura de tela feita durante uma execução de teste.

Capítulo 4

Resultados

4.1 Grupo de dados de teste para o modelo SVM

Para facilitar a compreensão dos resultados obtidos e criar um paralelo com o capítulo anterior, nesse capítulo divide-se os resultados em dois grandes conjuntos: o dos modelos gerado operando de maneira isolada e os resultados do modelo *Suport Vector Machine* operando como um nó da BSN. O primeiro conjunto de resultados refere-se aos testes aplicados nos modelos de *Machine Learning* (AM) desenvolvido para detecção de casos suspeitos de covid-19, no caso o modelo adotado SVM e o modelo testado NB. Já o segundo conjunto visa testar a detecção de casos suspeitos de covid-19 usando o nó "/covid_detection", onde foram traçados resultados para cada uma das três possibilidades de entrada disponíveis, sendo elas dados simulados, reais ou externos. Em suma, no primeiro conjunto o modelo de *Machine Learning* (AM) foi testado de maneira isolada e no segundo conjunto ele foi testado dentro da *Body Sensor Network* (BSN) junto a cada uma de suas possibilidades de coleta de dados.

4.2 Resultados do modelo de SVM para o conjunto de casos de teste

Anteriormente foi segregado um conjunto de dados de teste, correspondente a 30% do banco de dados total, contando com um conjunto para esses testes de 3.000 entradas. Esses dados foram então separados de seus respectivos resultados, para em seguida o modelo de predição treinado possa tentar prever se os dados inseridos pertencem ou não ao quadro de um paciente com covid-19. Com base nessas respostas, pode-se observar os resultados na Figura 4.1, que apresenta a matriz de confusão que compara os resultados previstos, com os esperados. Nessa figura, o eixo Y, à esquerda da matriz, representa

o resultado obtido pelo modelo de predição, e o X, na parte inferior, os resultados reais esperados. Em ambos, adota-se 0 para ”negativo para a suspeita de covid-19” e 1 para ”positivo para a suspeita de covid-19”.

Figura 4.1: Matriz de confusão com os resultados do modelo treinado aplicado aos dados de teste. ”1” representa casos suspeitos para covid-19 e ”0” os casos sem suspeita.

Nesse teste, a entrada contava com três rótulos. Para os 3.000 casos estudados, obteve-se a previsão correta para 2.986 casos e a previsão incorreta para 14 casos, afirmando assim uma acurácia 99,53% nos acertos para o conjunto de dados separados para teste.

4.3 Resultados do modelo de NB para o conjunto de casos de teste

De forma análoga ao caso anterior, o modelo desenvolvido usando o *Machine Learning* (AM) *Naïve Bayes* (NB). Nesse teste, mais uma vez, a entrada contava com três rótulos para os 3.000 casos estudados. Nesse modelo obteve-se a previsão correta para 2.675 casos e a previsão incorreta para 325 casos, resultando em uma acurácia de 89,17% nos acertos para o conjunto de dados separados para teste. Em função ao seu resultado inferior ao do modelo *Suport Vector Machine* (SVM) para o caso estudado, ele não foi usado nos demais testes.

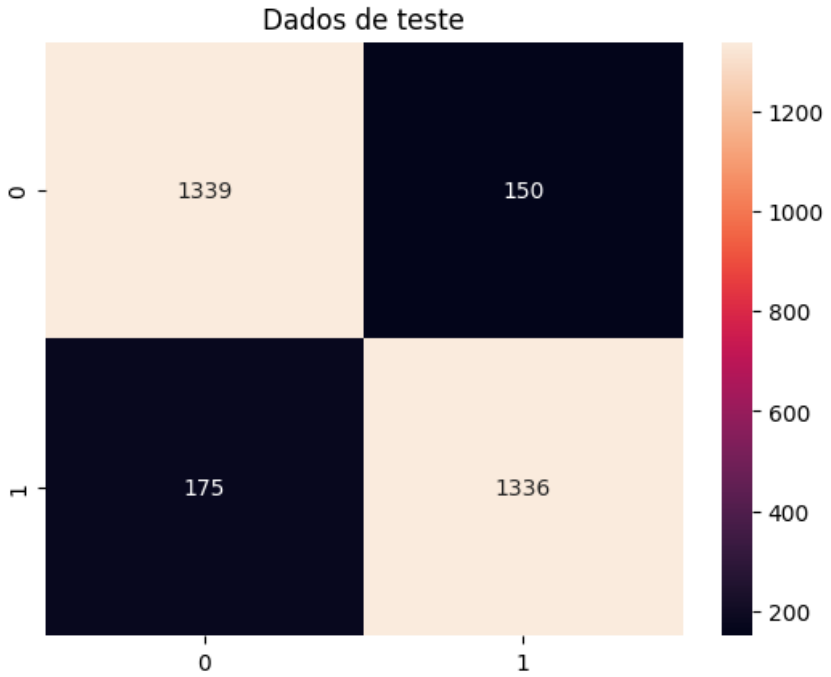


Figura 4.2: Matriz de confusão com os resultados do modelo NB treinado aplicado aos dados de teste. ”1” representa casos suspeitos para covid-19 e ”0” os casos sem suspeita.

4.4 Resultados de detecção de casos suspeitos de covid-19 com a BSN

Com a integração do modelo SVM de detecção de casos suspeitos de covid-19 junto a *Body Sensor Network*, constituindo um novo nó, foi possível usar todas as interfaces de entrada disponíveis como fonte de dados para a identificação de casos suspeitos de covid-19. Nos próximos tópicos existe o teste e resultados obtidos para cada tipo de entrada. Como os dados usados para fazer a análise de suspeita são os de temperatura, oxigenação e frequência cardíaca, apenas esses foram isolados. Para cada método de entrada será explicado o procedimento adotado e os resultados obtidos e esperados, e no capítulo seguinte esses resultados serão debatidos.

4.4.1 Resultado na detecção para dados simulados

Por ter sido adotada como coleta de dados, os dados simulados, no qual a geração é embasada na Cadeia de Markov, existiu uma menor coerência nos dados e portanto vale ressaltar que os resultados podem variar de execução para execução. Por, nesse caso, não existir uma relação direta nos dados gerados por cada sensor, ou seja, por cada um dos sensores gerar os próprios dados, os resultados desse teste se tornam não determinísticos,

e sim pseudo-aleatórios. A Cadeia de Markov adotada para esse teste foi a já configurada na versão base da BSN, estando essa representada no *Listing* 4.1. Nesse *Listing* pode-se observar dois conjuntos de dados para cada um dos seis sensores, o primeiro conjunto delimita os valores de cada faixa de risco e o segundo a probabilidade de transição entre cada uma delas. A cadeia a ser usada pode ser alterada para simular diferentes pacientes.

Listing 4.1: Codigo de configuração das Cadeias de Markov para os sensores usados na detecção de covid-19

```
<!-- Markov chain for oxigenation -->
<param name="oxigenation_State0" value="0,0,0,0,0" />
<param name="oxigenation_State1" value="0,0,0,0,0" />
<param name="oxigenation_State2" value="0,0,90,8,2" />
<param name="oxigenation_State3" value="0,0,75,10,5" />
<param name="oxigenation_State4" value="0,0,5,35,60" />

<!-- Risk values for oximeter -->
<param name="oxigenation_HighRisk0" value="-1,-1" />
<param name="oxigenation_MidRisk0" value="-1,-1" />
<param name="oxigenation_LowRisk" value="65,100" />
<param name="oxigenation_MidRisk1" value="55,65" />
<param name="oxigenation_HighRisk1" value="0,55" />

<!-- Markov chain for heart frequency -->
<param name="heart_rate_State0" value="72,21,4,2,1" />
<param name="heart_rate_State1" value="14,61,19,4,2" />
<param name="heart_rate_State2" value="1,17,60,20,2" />
<param name="heart_rate_State3" value="0,2,15,70,13" />
<param name="heart_rate_State4" value="0,1,2,20,77" />

<!-- Risk values for heart frequency -->
<param name="heart_rate_HighRisk0" value="0,70" />
<param name="heart_rate_MidRisk0" value="70,85" />
<param name="heart_rate_LowRisk" value="85,97" />
<param name="heart_rate_MidRisk1" value="97,115" />
<param name="heart_rate_HighRisk1" value="115,300" />

<!-- Markov chain for temperature -->
<param name="temperature_State0" value="25,51,21,3,0" />
```

```
<param name="temperature_State1" value="5,50,43,2,0" />
<param name="temperature_State2" value="0,4,85,11,0" />
<param name="temperature_State3" value="0,1,32,67,0" />
<param name="temperature_State4" value="0,0,0,0,0" />

<!-- Risk values for temperature -->
<param name="temperature_HighRisk0" value="0,31.99" />
<param name="temperature_MidRisk0" value="32,35.99" />
<param name="temperature_LowRisk" value="36,37.99" />
<param name="temperature_MidRisk1" value="38,40.99" />
<param name="temperature_HighRisk1" value="41,50" />
```

Para a Cadeia de Markov apresentada no *Listing*, foram obtidos os seguintes resultados representados na Tabela 4.1. O critério para essa coleta de casos foi selecionar os dados como simulados e executar a BSN coletando todas as primeiras ocorrências nas quais todas as leituras de cada um dos sensores isolados fossem diferentes da ocorrência anterior registrada. Resultando assim, em 50 casos registrados, dos quais 29 foram positivos e 21, negativos. Na Figura 4.3, pode-se observar, em diferentes modelos gráficos, a dispersão por dupla de sensores dos dados identificados como positivo, em laranja, e negativo, em azul, para suspeita de covid-19. Nessas representações visuais foram considerados dados de oxigênio, frequência cardíaca, temperatura corporal e a previsão feita pelo modelo.

Tabela 4.1: Tabela com os resultados obtidos nos casos de teste para o uso da BSN com entrada de dados simulados para detecção de casos suspeitos de covid-19. Legenda adotada: FC: Frequência Cardíaca e Obtido: Resultado Obtido

Caso	Oxigenação (%)	FC (BPM)	Temperatura (°C)	Obtido
1	95.23	96.02	38.66	Negativo
2	73.74	88.65	37.44	Positivo
3	77.60	90.52	37.90	Positivo
4	85.97	93.38	37.67	Positivo
5	73.41	89.08	39.74	Positivo
6	99.20	96.91	38.25	Negativo
7	90.73	94.01	37.77	Negativo
8	99.67	86.62	38.27	Negativo
9	69.10	94.85	36.54	Positivo
10	75.65	88.83	36.85	Positivo
11	80.37	70.74	36.31	Positivo

12	65.80	78.67	37.36	Positivo
13	99.70	83.47	38.00	Negativo
14	71.13	80.28	37.58	Positivo
15	68.99	73.10	36.63	Positivo
16	78.31	75.77	36.98	Positivo
17	72.21	77.15	37.17	Positivo
18	82.42	84.97	38.20	Positivo
19	98.58	83.08	37.94	Negativo
20	97.49	79.92	37.52	Negativo
21	91.10	73.11	36.62	Negativo
22	98.27	79.09	36.70	Negativo
23	79.90	76.02	37.01	Positivo
24	99.68	79.00	37.41	Negativo
25	76.14	80.07	37.55	Positivo
26	69.50	71.42	36.40	Positivo
27	91.10	77.33	37.19	Negativo
28	73.77	85.19	38.23	Positivo
29	94.08	78.22	38.04	Negativo
30	71.77	83.73	37.79	Positivo
31	76.22	73.30	36.66	Positivo
32	96.14	83.85	38.05	Negativo
33	88.87	75.87	36.99	Positivo
34	91.15	82.97	37.93	Negativo
35	67.46	75.15	36.89	Positivo
36	69.88	80.13	37.55	Positivo
37	68.42	79.05	37.41	Positivo
38	87.48	77.63	37.22	Positivo
39	83.82	76.69	37.10	Positivo
40	93.17	91.27	36.45	Negativo
41	95.92	89.98	36.90	Negativo
42	93.10	94.08	37.54	Negativo
43	91.09	94.77	36.53	Negativo
44	81.79	89.00	36.78	Positivo
45	75.74	86.99	36.81	Positivo
46	93.07	97.13	37.92	Negativo

47	76.25	95.06	36.48	Positivo
48	70.41	94.38	37.83	Positivo
49	94.01	91.10	36.44	Negativo
50	92.02	90.12	36.27	Negativo

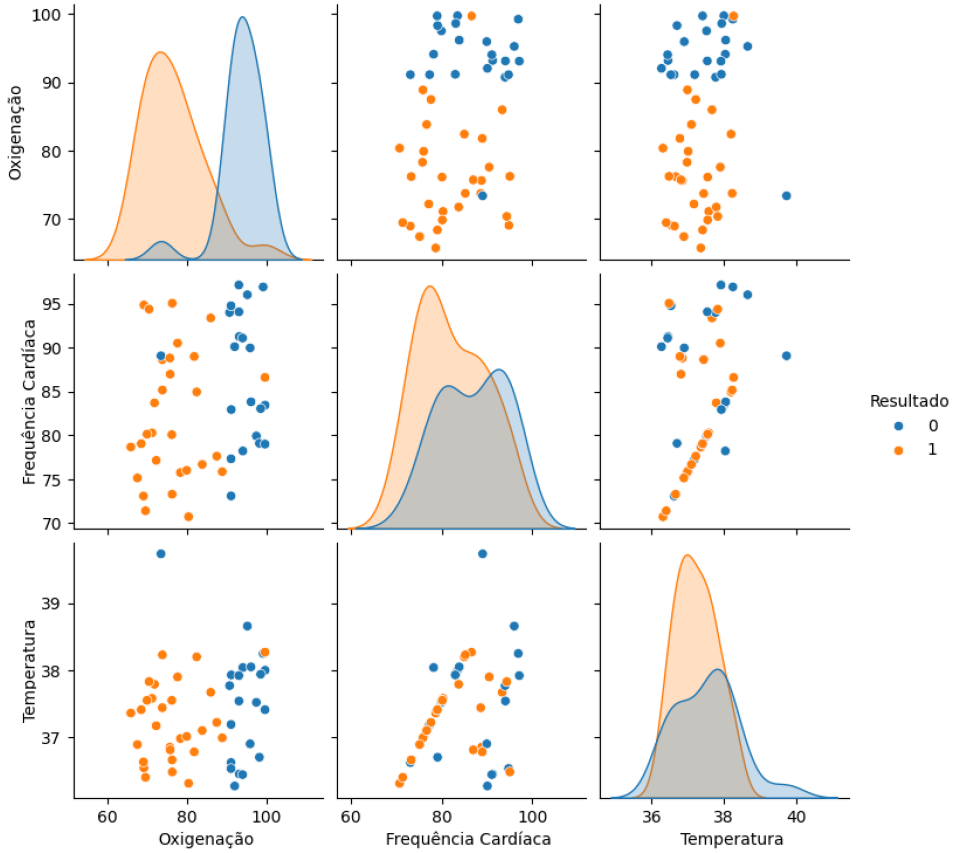


Figura 4.3: Gráfico com resultados da detecção de covid para dados simulados.

4.4.2 Resultado na detecção para dados externos

Como conjunto de dados determinísticos e de mais simples experimentação, usou-se os dados externos fornecidos pelo operador. Aqui foi possível o mapeamento dos dados de entrada e resultados esperados, conforme o apresentado na Tabela 4.2. Nessa tabela foram estabelecidos 10 casos de teste, nos quais em cinco se esperavam resultados negativos e nos outros cinco, resultados positivos para a suspeita de covid-19. Esse conjunto de casos de teste contou com casos extraídos do banco de dados inicial, mais precisamente do grupo separado para testes, usando como base um sorteio com o requisito de serem cinco casos positivos e cinco negativos.

Para esses testes foram feitas as seleções de entrada como dados externos, opção 2, para os três sensores: “g3t1_1”, de oxigenação na corrente sanguínea; “g3t1_2”, de frequência cardíaca; e “g3t1_3”, de temperatura. Também foram incluídos os respectivos arquivos CSV de dados na pasta “data_to_read”. Para cada rodada de teste foi executado o software com apenas os dados desejados, sendo feitas dez execuções, uma para cada conjunto de entrada.

Foram obtidos os seguintes resultados representados na Tabela 4.2. Comparando os resultados esperados e os obtidos, fica evidente uma taxa de acerto de 100%. Na Figura 4.4 pode-se observar a dispersão por dupla de sensores dos dados de frequência cardíaca, oxigenação e temperatura, identificados como positivo, em laranja, e negativo, em azul, para suspeita de covid-19. Para essa leitura também foi montado uma matriz de confusão apresentada na Figura 4.5, apresentando apenas resultados verdadeiros positivos e verdadeiros negativos.

Caso	Oxigenação (%)	FC (BPM)	Temperatura (°C)	Esperado	Obtido
1	98	106	38,33	Negativo	Negativo
2	94	56	35,56	Negativo	Negativo
3	95	79	36,11	Negativo	Negativo
4	96	59	36,67	Negativo	Negativo
5	100	58	37,22	Negativo	Negativo
6	91	127	36,67	Positivo	Positivo
7	92	124	35	Positivo	Positivo
8	89	101	36,11	Positivo	Positivo
9	86	108	37,22	Positivo	Positivo
10	92	42	36,11	Positivo	Positivo

Tabela 4.2: Tabela com os casos de teste para o uso da BSN com entrada de dados tabelada para detecção de casos suspeitos de covid-19 com os resultados esperados e obtidos. Legenda adotada: FC: Frequência Cardíaca, Esperado: Resultado Esperado e Obtido: Resultado Obtido

4.4.3 Resultado na detecção para dados reais coletados com sensores físicos

Ao se tratar da coleta de dados de dados reais, utilizou-se os sensores físicos para aferir o estado do paciente. Nesse estudo aplicou-se como cobaia apenas o autor, sendo esperado o resultado negativo para a suspeita de covid-19. Para realizar esse teste foram adotados os dois sensores físicos disponíveis, o MAX30102 e o DS18B20, fornecendo os dados usados para determinação de suspeita de covid-19. Para conduzir esse experimento o paciente segurou ambos os sensores, assim como apresentado na Figura 4.6. Foi então aguardado

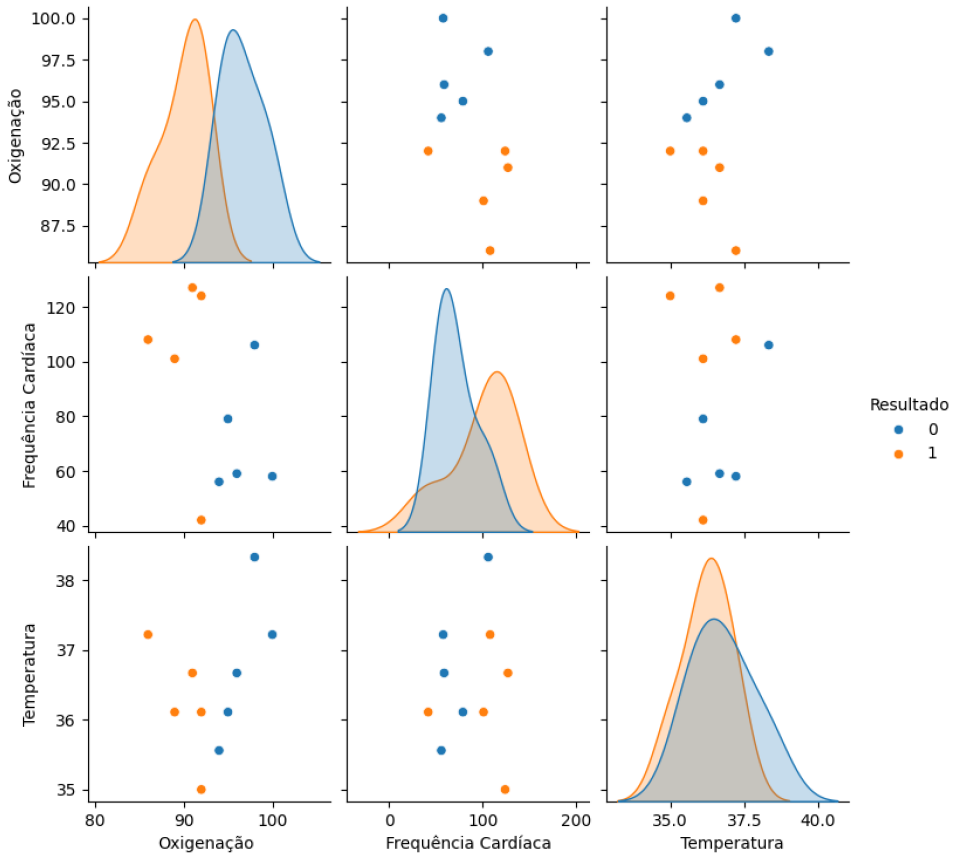


Figura 4.4: Gráfico com resultados da detecção de covid para dados externos.

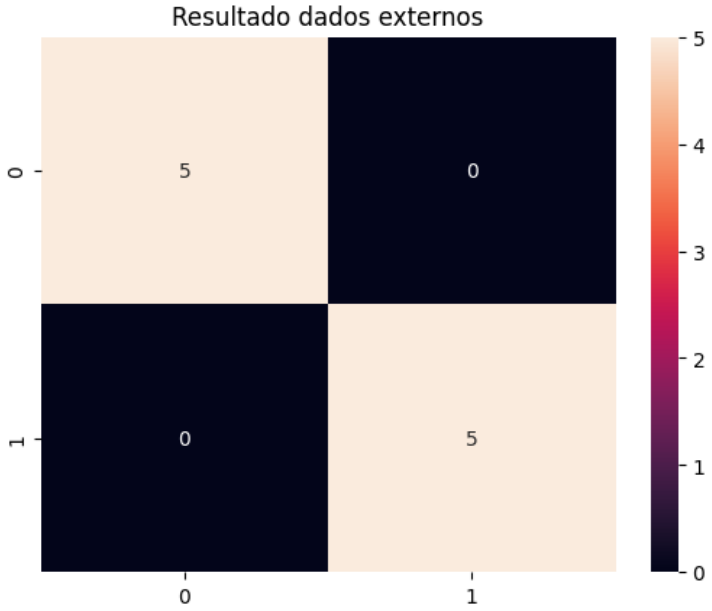


Figura 4.5: Matriz de confusão dos resultados da detecção de covid para dados tabelados.

até que os dados aferidos pelos sensores se mostrassem estáveis e então registrado o resultado, esse apresentado na Figura 4.7, se mostrou como não suspeita de covid-19, assim como esperado, uma vez que o paciente não apresentava diagnóstico positivo ou sintomas da doença por coronavírus.

Nessa Figura 4.7 é possível ver as mensagens do nó "/covid_detection" isoladas. Nela o nome do nó é apresentada na parte esquerda e na parte da direita a mensagem enviada por ele. Foi escolhido mostrar na figura os dados de "*Debug*", ou seja, um conjunto de dados extra geralmente não apresentado ao usuário. Nesses dados pode-se observar os valores aferidos pelos sensores físicos e a predição feita pelo modelo de *Machine Learning* (AM) para a suspeita da doença.

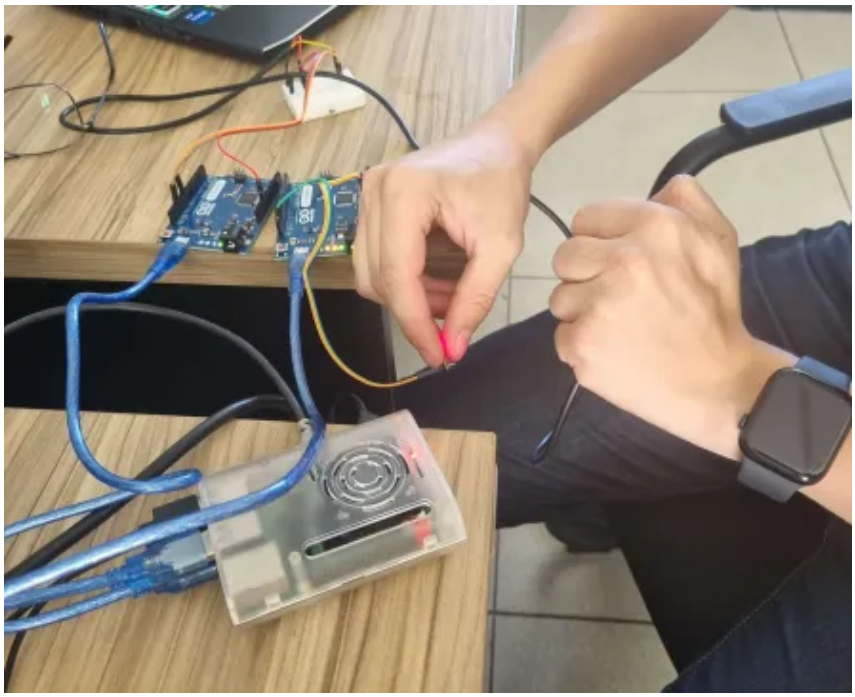


Figura 4.6: Paciente segurando os sensores físicos da BSN para aferir suspeita de covid-19.

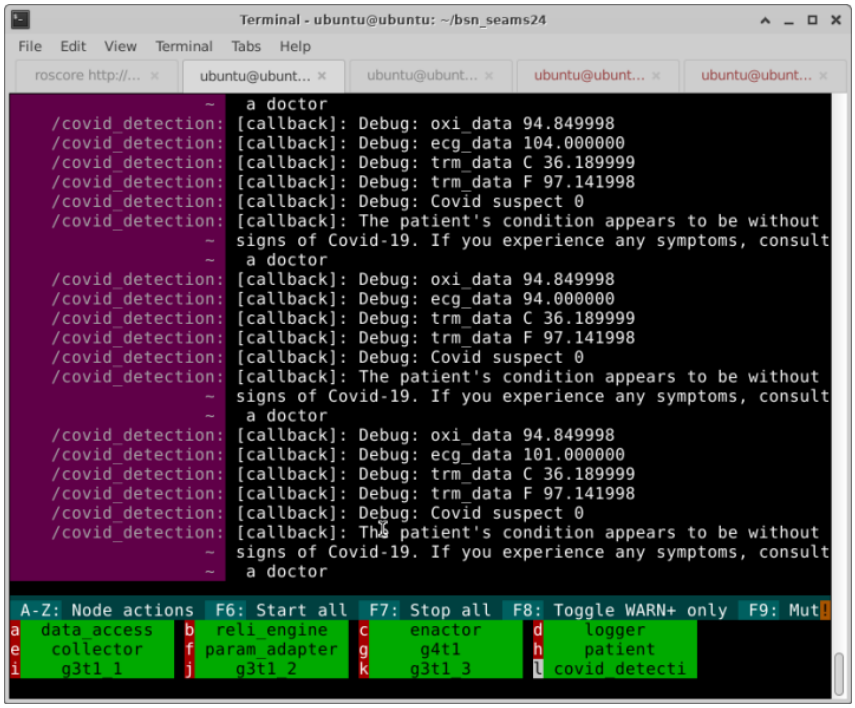


Figura 4.7: Resultado negativo para suspeita de covid-19 no paciente real usando sensores físicos.

Capítulo 5

Conclusões

5.1 Conclusões Gerais

A Tecnologia da Informação (TI) e engenharia avançam constantemente e são acompanhados pelos demais campos do conhecimento. Isso gera uma grande necessidade de construir ferramentas integradoras de áreas, provenientes de pesquisas interdisciplinares que agregam um conhecimento muito fragmentado.

Em um contexto mais específico, assim como dito por Rosalind Franklin, "A ciência e a vida cotidiana não podem e não devem ser separadas", um mundo pós pandêmico é marcado pela necessidade de preparar ferramentas para o próximo desafio incerto. Aqui estabeleceu-se um insumo interligador das áreas de computação e saúde, auxiliando no acompanhamento e diagnóstico de pacientes. O reconhecimento do covid-19 por uma *Body Sensor Network* é apenas uma abertura de portas para detecção de outras enfermidades.

Também não deve limitar essa ferramenta a sua casca física, o hardware, que é mutável e cada vez mais capilar. A popularização da tecnologia vestível, também conhecida como *wearables*, marca uma maior coleta de dados e possibilidade de pré diagnósticos em tempo real.

5.2 Conclusões para os testes

No decorrer desse trabalho, foram analisados e desenvolvidos quatro testes distintos: O teste do modelo de predição *Suport Vector Machine* (SVM) de detecção de casos suspeitos de covid-19 e o teste do uso da *Body Sensor Network* (BSN) para detecção de casos suspeitos de covid-19 realizado em três testes, um para cada possibilidade de entrada de dados.

Para o primeiro teste, o resultado se mostrou satisfatório para o modelo SVM, com uma acurácia de 99,53% nas predições para os 3.000 casos de teste. Já o modelo *Naïve*

Bayes (NB), por ter apresentado resultados inferiores ao anterior, não foi continuado para os próximos testes. Os testes relacionados a BSN se mostraram satisfatórios também, principalmente para os dados simulados e os dados externos, uma vez que todos os resultados esperados foram atingidos, independente da entrada de dados escolhida. Para o caso de dados reais o campo de pesquisa foi muito limitado, mas se mostrou promissor e com possibilidade de expansão no número de casos estudados.

5.3 Conclusões para as questões e objetivos de pesquisa

Inicialmente foram elaboradas três questões de pesquisa 1.3:

- Qual é o papel da *Body Sensor Network* (BSN) na detecção de casos de covid-19?
  - Os resultados encontrados no decorrer desse estudo são incipientes mas muito positivos. Não apenas foi possível realizar a detecção dos casos com o uso da BSN, como também foi possível adotar diferentes interfaces de entradas de dados para fazer isso.
- Como especializar uma BSN para detecção de covid-19?
  - O cerne desse trabalho foi captar os dados do paciente e processá-los da maneira acurada. Aqui foi optado o uso de três interfaces de entrada e um processamento usando um modelo de aprendizagem *Suport Vector Machine* (SVM), mas outras entradas e modelos podem ser adotados. As entradas de dados se mostraram viáveis e eficientes, enquanto o modelo adotado se mostrou acurado e eficiente para todos os casos de teste adotados.
- O que desenvolvedores de uma *Body Sensor Network* devem considerar na especialização de uma BSN genérica para detecção de covid-19?
  - É preciso uma atenção especial no entendimento do trabalho base adotado, vez que esse pode já possuir mecanismos de processamento de dados pré implantados e, principalmente, manipulações de dados como filtros e injetores de incerteza que podem ajudar ou atrapalhar a conquista dos resultados esperados.

Se referindo aos objetivos estabelecidos inicialmente, todos eles foram cumpridos de maneira sequencial, orientando o bom desenvolvimento desse trabalho e garantindo sua adequação à proposta inicialmente estabelecida.

5.4 Trabalhos Futuros

Algumas linhas de estudo devem ser consideradas para gerar uma continuidade ao presente trabalho, podendo tanto melhorar as funcionalidades implementadas quanto desenvolver novas:

5.4.1 Coletar dados de usuários reais

O uso dos dados disponíveis para treinar o modelo de *Machine Learning* (AM) se limitou a dados mistos entre gerados e reais. O uso da *Body Sensor Network* para coletar dados reais e a busca de bancos de dados com pacientes reais diagnosticados e de controle seria de grande valor. Como prova de conceito os dados adotados se mostraram satisfatórios, entretanto, como aplicação real, a busca de dados mais condizentes com o cotidiano seria imprescindível.

5.4.2 Desenvolver um modelo com uma análise temporal, e não pontual

A atual detecção da enfermidade, no caso o covid-19, se limita a uma análise pontual do estado do paciente. O uso de uma base de dados de treino que envolva um acompanhamento mais intenso e contínuo, provavelmente traria resultados mais acurados e significativos.

5.4.3 Integração com tecnologias já presentes no mercado

Com a popularização de tecnologias vestíveis, tais como *smartwatches*, a coleta e processamento de dados fisiológicos se torna mais corriqueira e acessível. Existe nesse meio a possibilidade de integrar a *Body Sensor Network* (BSN) a opções de *harware* disponíveis no mercado que já fazem a coleta e alguns processamentos na área de saúde. A busca de parceiros estratégicos e criação de aplicativos compatíveis com grandes fabricantes no mercado pode se mostrar valoroso para aplicar a tecnologia aqui empregada em larga escala.

5.4.4 Adoção de mais sensores e coleta de dados

Com o intuito de permitir uma maior gama de diagnósticos, inclusive para outras enfermidades, e aprimorar a confiabilidade do sistema, a adição de sensores e coletas de dados se mostra promissora. A possibilidade de inclusão de sintomas do paciente, tais como calafrios, coceira, irritação ou coriza, pode contribuir significativamente para o

diagnóstico. Adotar mais sensores fisiológicos também pode ser significativo, sensores tais como de glicose, de monóxido de carbono e de atividade muscular. A inclusão de dados comuns, como peso, idade e altura, pode ajudar em análises mais elaboradas.

5.4.5 Treinamento para detecção de outras enfermidades

Criar novos nós com novos modelos de *Machine Learning* é um caminho viável para expandir a funcionalidade da BSN, implantando a detecção de novas enfermidades e tornando o diagnóstico do paciente mais completo.

5.4.6 Adoção de sensores mais estáveis e acurados

Durante o uso dos sensores físicos MAX30102 e DS18B20, principalmente o MAX30102 se mostrou pouco acurado e estável. Por esse motivo os dados aferidos se mostraram distantes dos reais e dificultaram a detecção de casos de covid-19 e o sensor precisou ser reiniciado algumas vezes durante alguns dos testes. Para resolver isso pode-se buscar sensores semelhantes mais estáveis e acurados.

Referências

[1] Gil, Eric Bernd, Ricardo Caldas, Arthur Rodrigues, Gabriel Levi Gomes da Silva, Genaína Nunes Rodrigues e Patrizio Pelliccione: *Body sensor network: A self-adaptive system exemplar in the healthcare domain.* Em *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, páginas 224–230, 2021. x, 1, 2, 3, 8, 11, 14

[2] VOLP: *covid-19, covid-19 - vocabulário ortográfico da língua portuguesa.* `https://www.volp-acl.pt/index.php/item/covid-19`, acesso em 17-12-2023. 1

[3] Awotunde, Joseph Bamidele, Rasheed Gbenga Jimoh, Muyideen AbdulRaheem, Idowu Dauda Oladipo, Sakinat Oluwabukonla Folorunso e Gbemisola Janet Ajamu: *IoT-Based Wearable Body Sensor Network for COVID-19 Pandemic*, páginas 253–275. Springer International Publishing, Cham, 2022, ISBN 978-3-030-77302-1. 1

[4] Lin, Kai, Yihui Li, Jinchuan Sun, Dongsheng Zhou e Qiang Zhang: *Multi-sensor fusion for body sensor network in medical human–robot interaction scenario.* Information Fusion, 57:15–26, 2020, ISSN 1566-2535. 1

[5] Li, Yujie, Huimin Lu, Hyoungseop Kim e Seiichi Serikawa: *Touch switch sensor for cognitive body sensor networks.* COMPUTER COMMUNICATIONS, 146:32–38, OCT 15 2019, ISSN 0140-3664. 1

[6] Wang, Yifei, Huizhi Wang, Jin Xuan e Dennis Y.C. Leung: *Powering future body sensor network systems: A review of power sources.* Biosensors and Bioelectronics, 166:112410, 2020, ISSN 0956-5663. 1, 7

[7] Laurijssen, Dennis, Wim Saeys, Steven Truijen, Walter Daems e Jan Steckel: *Synchronous wireless body sensor network enabling human body pose estimation.* IEEE ACCESS, 7:49341–49351, 2019, ISSN 2169-3536. 1

[8] Hirten, Robert P, Matteo Danieletto, Lewis Tomalin, Katie Hyewon Choi, Micol Zweig, Eddye Golden, Sparshdeep Kaur, Drew Helmus, Anthony Biello, Renata Pyzik, Alexander Charney, Riccardo Miotto, Benjamin S Glicksberg, Matthew Levin, Ismail Nabeel, Judith Aberg, David Reich, Dennis Charney, Erwin P Bottinger, Laurie Keefer, Mayte Suarez-Farinas, Girish N Nadkarni e Zahi A Fayad: *Use of physiological data from a wearable device to identify sars-cov-2 infection and symptoms and predict covid-19 diagnosis: Observational study.* J Med Internet Res, 23(2):e26107, Feb 2021, ISSN 1438-8871. 1

[9] Sanghvi, Harshal A., Riki H. Patel, Ankur Agarwal, Shailesh Gupta, Vivek Sawhney e Abhijit S. Pandya: *A deep learning approach for classification of covid and pneumonia using densenet-201.* International journal of imaging systems and technology, 33(1):18–38, 2023, ISSN 0899-9457. 1

[10] Ali, Shokat, Ravi Pratap Singh, Mohd Javaid, Abid Haleem, Honey Pasricha, Rajiv Suman e Jimmy Karloopia: *A review of the role of smart wireless medical sensor network in covid-19.* Journal of Industrial Integration and Management, 05(04):413–425, 2020. https://doi.org/10.1142/S2424862220300069. 2

[11] Floos, Ahmad Yahya M. e Ahmad S. Al-Mogren: *Smart bodynet for hypercube body sensor network.* Em *2015 5th National Symposium on Information Technology Towards New Smart World (NSITNSW)*, 2015, ISBN 978-1-4799-7627-0. 2

[12] Goodman, Elizabeth e Patricia Henry: *Product methodologies - what they are and how to avoid pitfalls - pmi.* https://www.pmi.org/learning/library/product-methodologies-software-development-programs-6529. 3

[13] Robotics, Open: *Ros.org | powering the world's robots.* https://www.ros.org/, acesso em 26-11-2023. 5

[14] ros.org: *Um pouco mais sobre nós("nodes") no ros.* http://wiki.ros.org/pt_BR/ROS/Tutorials/UnderstandingNodes, acesso em 23-12-2023. 5, 6

[15] Li, W. e C. Zhang: *Markov chain analysis.* Em Kitchin, Rob e Nigel Thrift (editores): *International Encyclopedia of Human Geography*, páginas 455–460. Elsevier, Oxford, 2009, ISBN 978-0-08-044910-4. 6

[16] University, Columbia e Mailman School of Public Health: *Markov chain monte carlo | columbia public health.* https://www.publichealth.columbia.edu/research/population-health-methods/markov-chain-monte-carlo, acesso em 26-11-2023. 6

[17] Caldas, Ricardo Diniz: *Prototipação e verificação formal de sistema autônomo com propriedades tempo-real : um estudo de caso no body sensor network.* Biblioteca Digital da Produção Intelectual Discente, 57, 2017. 7, 8

[18] LastMinuteEngineers: *Interfacing max30102 pulse oximeter and heart rate sensor with arduino.* https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/, acesso em 26-11-2023. 10

[19] Elprocus: *Ds18b20 temperature sensor : Pin diagram, working & its applications.* https://www.elprocus.com/ds18b20-temperature-sensor/, acesso em 26-11-2023. 11

[20] Melo Peixoto, João Vitor de: *Aplicação de sistemas autoadaptativos em robótica móvel : um estudo de caso no robô pioneer.* Biblioteca Digital da Produção Intelectual Discente, 10483(28472), 2019. 16

[21] Júnior, Moacyr Gonçalves Cereja: *Um framework para adaptação dinâmica de software aplicado a sistemas de segmento solo*. Instituto Nacional de Pesquisas Espaciais - INPE, 2018. 16

[22] Gama, Juliana Karl Araujo Hiago Santos da e Aline Pires Vieira de Vasconcelos Juliana de Souza Monteiro, Rogerio Atem de Carvalho: *Especificação de requisitos para sistemas embarcados: Uma revisão sistemática da literatura*. Brazilian Journal of Development, 7(3):25216–25226, 2021. 17

[23] Arduino: *Arduino*. `https://www.arduino.cc/`, acesso em 26-11-2023. 17, 37

[24] Alisher Shakirovich Ismailov, Zafar Botirovich Jo'rayev: *Study of arduino microcontroller board*. Science and Education" Scientific Journal, 3(3):172–179, 2022. 17

[25] Arduino: *Arduino leonardo*. `https://docs.arduino.cc/hardware/leonardo`, acesso em 09-12-2023. 18

[26] Foundation, Raspberry Pi: *Raspberry pi*. `https://www.raspberrypi.com/`, acesso em 26-11-2023. 19

[27] Ana Carolina Lorena, André C. P. L. F. de Carvalho: *Uma introdução às support vector machines*. Revista de Informática Teórica e Aplicada, 2(14), 2017. 19, 20

[28] Cervantes, Jair, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua e Asdrubal Lopez: *A comprehensive survey on support vector machine classification: Applications, challenges and trends*. Neurocomputing, 408:189–215, 2020, ISSN 0925-2312. 20

[29] Amasifuen, Francisco Sebastian Tacora: *Uso de aprendizado de máquinas para reconhecimento de padrões*. Instituto de Matemática e Estatística, Universidade Federal Fluminense, 2021. 20

[30] Himani Bhavsar, Mahesh H. Panchal: *A review on support vector machine for data classification*. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2012. 20

[31] Chen, Jin, Cheng Wang e Runsheng Wang: *Adaptive binary tree for fast svm multiclass classification*. Neurocomputing, 72(13):3370–3375, 2009, ISSN 0925-2312. Hybrid Learning Machines (HAIS 2007) / Recent Developments in Natural Computation (ICNC 2007). 20

[32] Panagopoulos, Orestis P., Petros Xanthopoulos, Talayeh Razzaghi e Onur Seref: *Relaxed support vector regression*. ANNALS OF OPERATIONS RESEARCH, 276(1-2, SI):191–210, MAY 2019, ISSN 0254-5330. 20

[33] Mansour, Nehal A., Ahmed I. Saleh, Mahmoud Badawy e Hesham A. Ali: *Accurate detection of covid-19 patients based on feature correlated naïve bayes (fcnb) classification strategy*. Journal of Ambient Intelligence and Humanized Computing, 13(1):41–73, Jan 2022, ISSN 1868-5145. `https://doi.org/10.1007/s12652-020-02883-2`. 21

[34] SA-BSN: *Self-Adaptive Body Sensor Network Repository from the Software Engineering Lab (LES) @UnB*. https://github.com/lesunb/bsn, acesso em 09-12-2023. 33

[35] PyPI: *pandas 2.1.4*. https://pypi.org/project/pandas/, acesso em 16-12-2023. 42

[36] PyPI: *scikit-learn 1.3.2*. https://pypi.org/project/scikit-learn/, acesso em 16-12-2023. 42

[37] Matplotlib: *Matplotlib: Visualization with python*. https://matplotlib.org/, acesso em 16-12-2023. 42

[38] documetation, Python: *Pickle — python object serialization*. https://docs.python.org/3/library/pickle.html, acesso em 16-12-2023. 42

Anexo I

Código para treinamento do modelo *Suport Vector Machine* (SVM) de identificação de casos suspeitos de covid-19

Listing I.1: Código para treinamento do modelo SVM de detecção de casos suspeitos de covid-19 - Parte 1 de 3

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.
    read_csv)

import seaborn as sns
import pickle
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
%matplotlib inline

df = pd.read_csv('input/dataset-covid-csv/dataset_covid_trad.
    csv')

# Substitui as palavras Positivo e Negativo por 1 e 0
df['Resultado'].replace(['Positivo','Negativo'],[1,0],inplace
    =True)
```

```
# Variavel para decidir se queremos apresentacoes de debug
debug = 0
if debug == 1:
    # Apresenta as primeiras 5 linhas dos dados
    print(df.head())
    # Apresenta a estrutura do dataframe
    print(df.shape)
    # Apresenta a quantidade de NaN em cada coluna
    print(df.isna().sum())
# define a coluna ID como o indice do df
df.set_index('ID', inplace=True)

# Apresenta a distribuicao inicial dos dados
sns.pairplot(data=df, hue='Resultado')
plt.show()
```

Listing I.2: Código para treinamento do modelo SVM de detecção de casos suspeitos de covid-19 - Parte 2 de 3

```
# Separa os dados dos resultados
dados_sem_result = df.drop('Resultado',axis=1) #x
dados_result = df['Resultado'] #y

# Divide os dados em 30% para teste e 70% para treinamento
dados_sem_result_train, dados_sem_result_test,
   dados_result_train, dados_result_test = train_test_split(
   dados_sem_result, dados_result, test_size=0.3)
# Cria uma instancia do classificador SVM (Suport Vector
   Machine)  Escolhemos o kernel rbf (Radial Basis Function)
model = SVC(C=5,gamma='auto')
# Este comando treina o modelo SVM nos dados de treinamento.
   A funcao fit() ajusta o modelo aos dados
model.fit(dados_sem_result_train, dados_result_train)

# Salvar o modelo treinado como um arquivo pickle
with open('model_covid_prediction.pkl', 'wb') as file:
    pickle.dump(model, file)
```

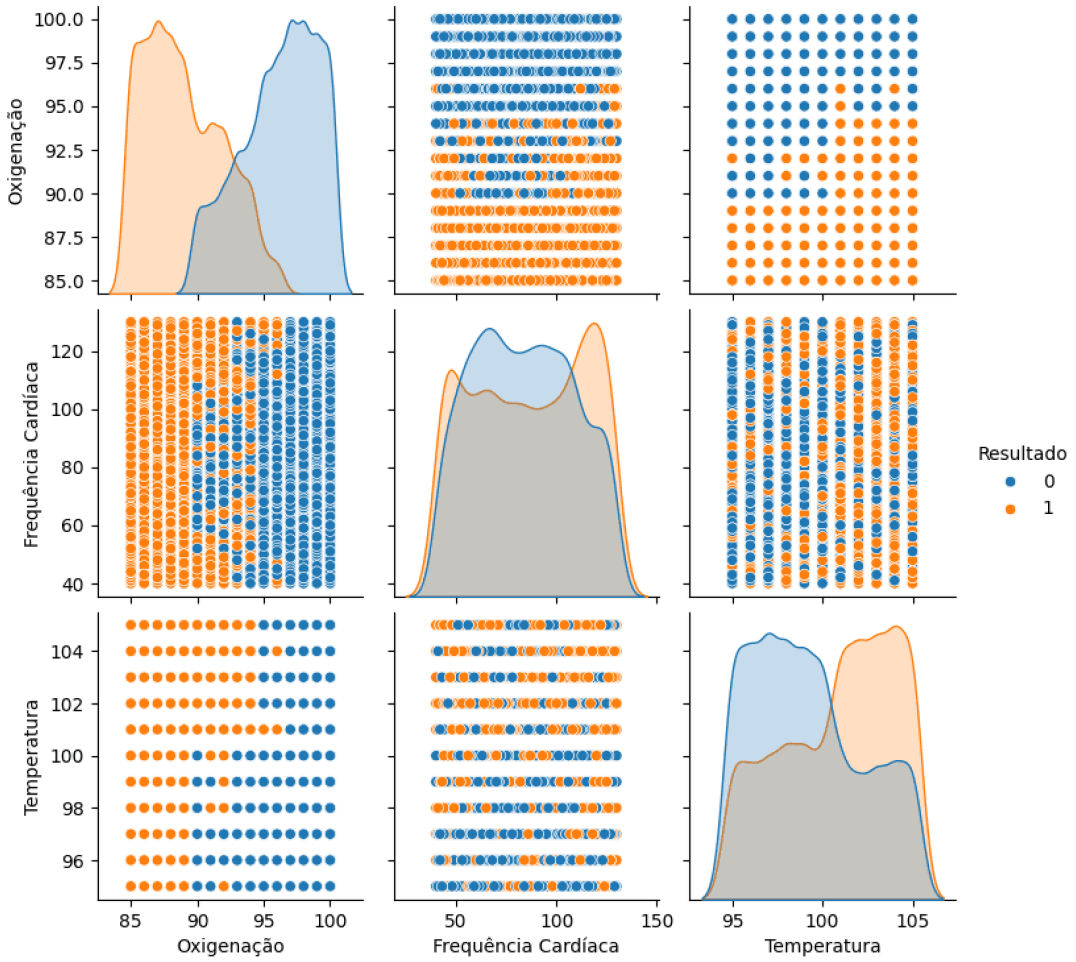


Figura I.1: Conjunto de gráficos representando a entrada de dados usada para treinamento e teste.

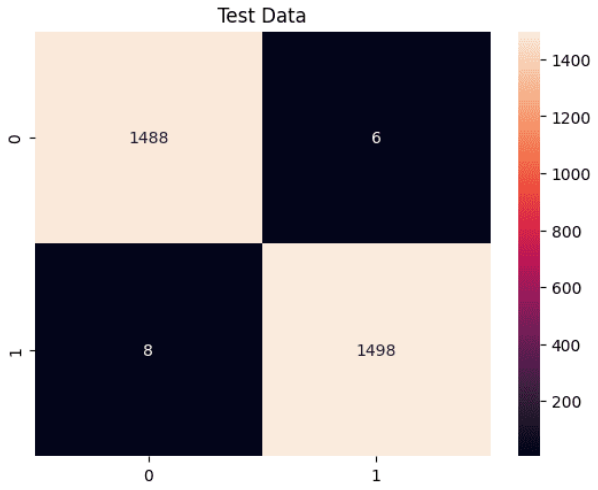


Figura I.2: Matriz de confusão com resultados do modelo SVM treinado aplicado aos dados de teste.

```
# Calcula a acuracia do modelo SVM nos dados de teste
print(model.score(dados_sem_result_test, dados_result_test))
# Calcula a acuracia do modelo SVM nos dados de treinamento
print(model.score(dados_sem_result_train, dados_result_train)
    )

dados_result_pred = model.predict(dados_sem_result_test)
y_pred_train = model.predict(dados_sem_result_train)
test = confusion_matrix(dados_result_pred, dados_result_test)

# Apresenta os dados de teste em um grafico
plt.title('Dados de teste')
sns.heatmap(data=test, annot=True, fmt='g')
plt.show()
```

Saída obtida:

0.9953333333333333

1.0

Listing I.3: Código para treinamento do modelo SVM de detecção de casos suspeitos de covid-19 - Parte 3 de 3

```
train = confusion_matrix(y_pred_train, dados_result_train)
```

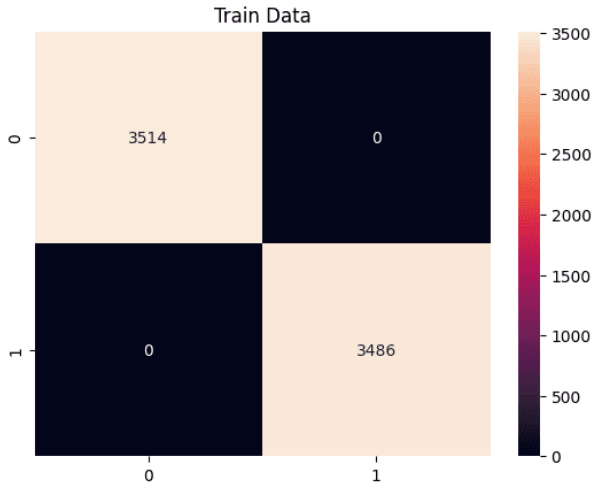


Figura I.3: Matriz de confusão com resultados do modelo SVM treinado aplicado aos dados de treinamento.

```
# Apresenta os dados de treinamento em um grafico
plt.title('Dados de treinamento')
sns.heatmap(data=train,annot=True,fmt='g')
plt.show()
```

Anexo II

Código para treinamento do modelo *Naïve Bayes* (NB) de identificação de casos suspeitos de covid-19

Listing II.1: Código para treinamento do modelo NB de detecção de casos suspeitos de covid-19 - Parte 1 de 3

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.
    read_csv)

import seaborn as sns
import pickle
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
%matplotlib inline

df = pd.read_csv('input/dataset-covid-csv/dataset_covid_trad.
    csv')

# Substitui as palavras Positivo e Negativo por 1 e 0
df['Resultado'].replace(['Positivo','Negativo'],[1,0],inplace
    =True)

# Variavel para decidir se queremos apresentacoes de debug
```

```
debug = 0
if debug == 1:
    # Apresenta as primeiras 5 linhas dos dados
    print(df.head())
    # Apresenta a estrutura do dataframe
    print(df.shape)
    # Apresenta a quantidade de NaN em cada coluna
    print(df.isna().sum())
# define a coluna ID como o indice do df
df.set_index('ID', inplace=True)

# Apresenta a distribuicao inicial dos dados
sns.pairplot(data=df, hue='Resultado')
plt.show()
```

Listing II.2: Código para treinamento do modelo NB de detecção de casos suspeitos de covid-19 - Parte 2 de 3

```
# Separa os dados dos resultados
dados_sem_result = df.drop('Resultado',axis=1) #x
dados_result = df['Resultado'] #y

# Divide os dados em 30% para teste e 70% para treinamento
dados_sem_result_train, dados_sem_result_test,
   dados_result_train, dados_result_test = train_test_split(
   dados_sem_result, dados_result, test_size=0.3)

# Cria uma instancia do classificador Naive Bayes
from sklearn.naive_bayes import GaussianNB
model = GaussianNB(var_smoothing=1e-25)

# Este comando treina o modelo Naive Bayes nos dados de
   treinamento. A funcao fit() ajusta o modelo aos dados
model.fit(dados_sem_result_train, dados_result_train)

# Salvar o modelo treinado como um arquivo pickle
with open('model_covid_prediction.pkl', 'wb') as file:
    pickle.dump(model, file)
```

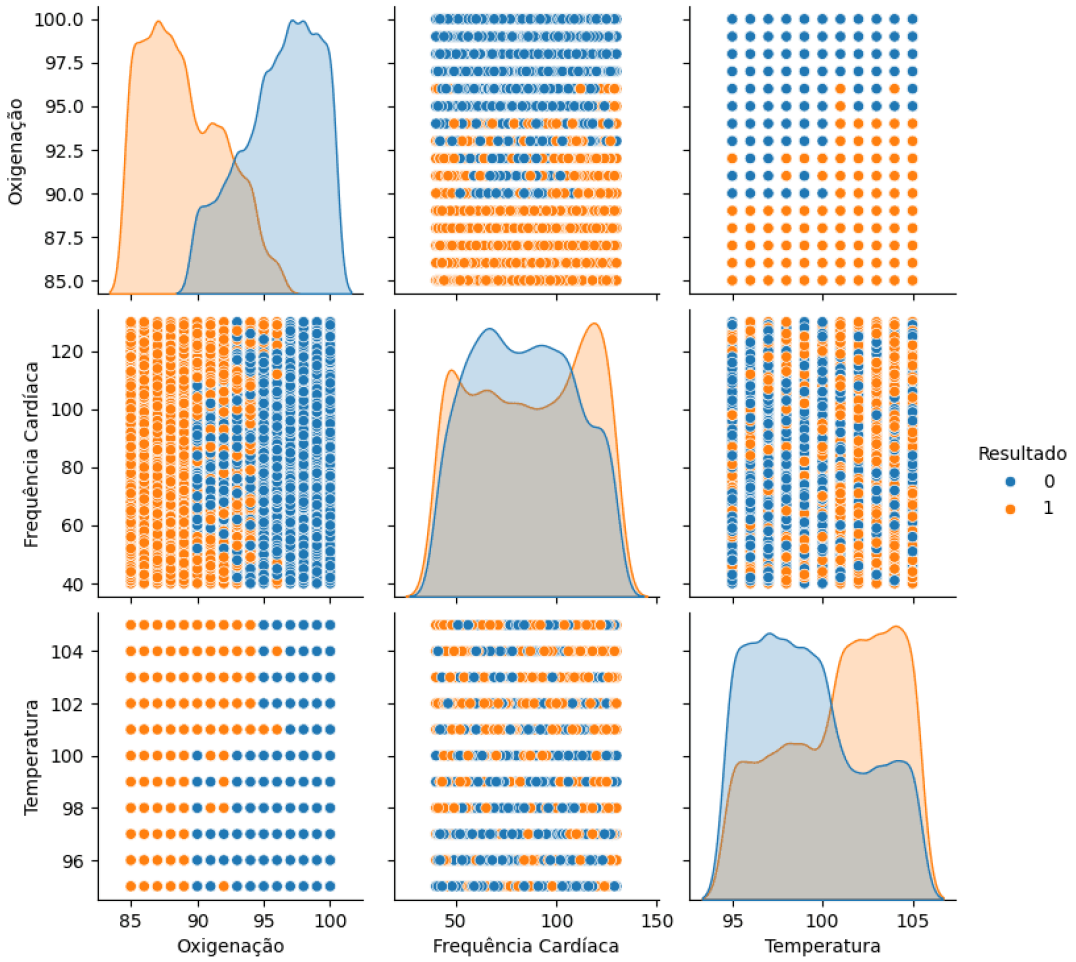


Figura II.1: Conjunto de gráficos representando a entrada de dados usada para treinamento e teste.

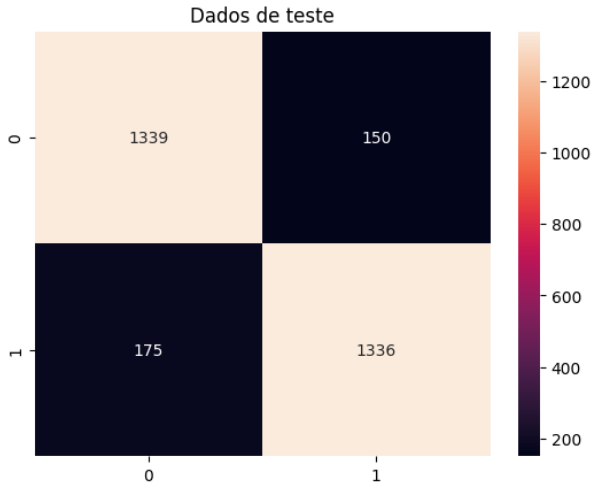


Figura II.2: Matriz de confusão com resultados do modelo *Naïve Bayes* treinado aplicado aos dados de teste.

```
# Calcula a acuracia do modelo Naive Bayes nos dados de teste
print(model.score(dados_sem_result_test, dados_result_test))
# Calcula a acuracia do modelo Naive Bayes nos dados de
    treinamento
print(model.score(dados_sem_result_train, dados_result_train)
    )

dados_result_pred = model.predict(dados_sem_result_test)
y_pred_train = model.predict(dados_sem_result_train)
test = confusion_matrix(dados_result_pred, dados_result_test)

# Apresenta os dados de teste em um grafico
plt.title('Dados de teste')
sns.heatmap(data=test, annot=True, fmt='g')
plt.show()
```

Saída obtida:
0.8916666666666667
0.8875714285714286

Listing II.3: Código para treinamento do modelo SVM de detecção de casos suspeitos de covid-19 - Parte 3 de 3

```
train = confusion_matrix(y_pred_train, dados_result_train)
```

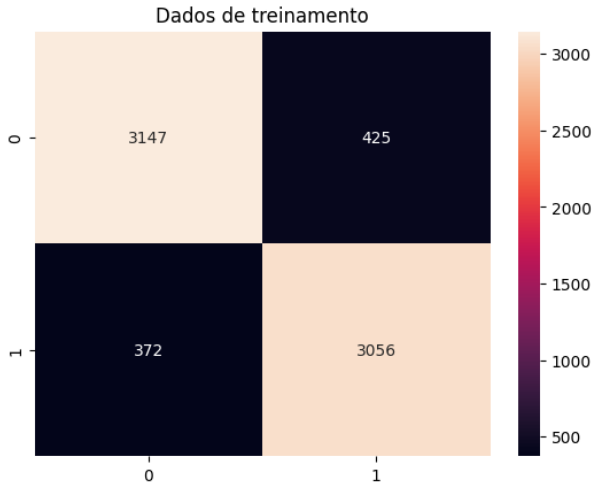


Figura II.3: Matriz de confusão com resultados do modelo *Naïve Bayes* treinado aplicado aos dados de treinamento.

```
# Apresenta os dados de treinamento em um grafico
plt.title('Dados de treinamento')
sns.heatmap(data=train,annot=True,fmt='g')
plt.show()
```

Anexo III

Código do nó covid_detection para predição de covid-19

Listing III.1: Código em python usado para criar um nó na BSN usado para detectar casos suspeitos de covid-19

```
#!/usr/bin/env python
import rospy
from messages.msg import TargetSystemData

import pickle
import pandas as pd

def callback(data):
    caminho = __file__.replace("covid_detection.py", "
       model_covid_prediction.pkl")
    # Abrir o arquivo pickle e carregar o modelo
    with open(caminho, 'rb') as file:
        model = pickle.load(file)

    data_to_test = pd.DataFrame({'Oxygen': data.oxi_data, '
       PulseRate': data.ecg_data, 'Temperature': ((data.
       trm_data * 1.8) + 32)}, index=[0])
    resultado = model.predict(data_to_test)
    debug = 1
    if(debug==1):
        # Realiza a previsao de um unico caso de teste
```

```
        rospy.loginfo("Debug: oxi_data %f" % (data.oxi_data))
        rospy.loginfo("Debug: ecg_data %f" % (data.ecg_data))
        rospy.loginfo("Debug: trm_data C %f" % (data.trm_data
            ))
        rospy.loginfo("Debug: trm_data F %f" % ((data.
            trm_data * 1.8) + 32))
        rospy.loginfo("Debug: Covid suspect %i" % (resultado)
            )
    if(resultado[0]==1):
        rospy.loginfo("The patient's condition was compatible
             with a suspected case of Covid-19. Please see a
            doctor.")
    else:
        rospy.loginfo("The patient's condition appears to be
            without signs of Covid-19. If you experience any
            symptoms, consult a doctor")

def listener():
    rospy.init_node('covid_detection_listener', anonymous=
        True)
    rospy.Subscriber("TargetSystemData", TargetSystemData,
        callback)

    # spin() simply keeps python from exiting until this node
         is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```